SCHAUM'S ouTlines

全美经典学习指导系列

数字系统导论

〔美〕J. 帕尔默 D. 帕尔曼 著

陈文楷 徐萍萍 译

- 涵盖全部课程基础
- 400多道精选习题及其详解
- 迅速提高解题能力
- 大大节省学习时间
- 自学的最佳参考书

科学出版社

麦格劳-希尔教育出版集团

全美经典学习指导系列

数字系统导论

〔美〕 J. 帕尔默
D. 帕尔曼 著

陈文楷 徐萍萍 译

科学出版社

麦格劳-希尔教育出版集团

2002

内 容 简 介

本书为数字电路的辅导教材，主要介绍能有效解决数字电路问题的一些实用设计方法。全书共 11 章，主要包括组合逻辑的设计及其方法、中、大规模集成电路基础，触发器及其组合专用器件，简单状态机的设计和电可编程功能。书中各章均先讲述一些基本概念和要点；之后介绍中大规模集成电路等的组合逻辑元件，重点讲述可满足特定应用的可编程元件；最后主要讨论同步时序逻辑电路。

本书适用于高等院校本科电气工程、电子工程、自动控制、计算机及相关专业，也可以作为老师教学和学生自学用参考书。

James Palmer, David Perlman: Schaum's Outlines Introduction to Digital Systems

ISBN: 0-07-048439-2

Copyright © 1993 by the McGraw-Hill Companies, Inc.

Authorized translation from the English language edition published by McGraw-Hill Companies, Inc.

All rights reserved.

本书中文简体字版由科学出版社和美国麦格劳-希尔教育出版集团合作出版，未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

版权所有，翻印必究。

本书封面贴有 McGraw-Hill 公司防伪标签，无标签者不得销售。

图字：01-2001-1775 号

图书在版编目(CIP)数据

数字系统导论/[美]J. 帕尔默等著；陈文楷，徐萍萍译. —北京：科学出版社，2002

(全美经典学习指导系列)

ISBN 7-03-009049-7

Ⅰ. 数… Ⅱ. ①帕… ②陈… ③徐… Ⅲ. 数字电路 Ⅳ. TN79

中国版本图书馆 CIP 数据核字（2001）第 074092 号

科 学 出 版 社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

丽源印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2002 年 1 月第 一 版 开本：A4（890×1240）

2002 年 1 月第一次印刷 印张：19

印数：1—5 000 字数：543 000

定价：28.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

前　言

本书的目的是在数字系统入门课程中介绍一般性的设计方法。这门课是以“数字系统导论”为基础，为罗切斯特理工学院学习电工技术课程的一年级学生和即将开始该课程学习的学生而开设。

本书第1章简要地介绍数制，重点介绍二进制，将其作为介绍二值逻辑变量以及所有计算机和数字电路中信号的基础。

接下来的3章介绍与组合逻辑系统相关的设计过程。第2章给出了组合逻辑设计问题的3种不同方法：真值表、布尔表达式和逻辑图。第3章论述布尔代数的运算，其中包括与卡诺图的结构和描述有关的内容。在此，设计问题就是在纯逻辑电平上的分析，不只考虑硬件，还探讨卡诺图、布尔表达式和逻辑图之间的关系。第4章使用混合逻辑方法，给出了逻辑的硬件实现的结构化方法，从而不仅得到了能产生功能性逻辑电路的非常清晰的设计工具，还保持了与最初内在的布尔关系的一致性。

第5章给出了通用MSI和LSI组合逻辑元件的说明，重点放在专用的可编程器件上(例如多路选择器和ROM)。

本书的其余部分是关于同步时序逻辑的内容。第6章逐步阐明时序逻辑电路的结构和时序波形图的使用，这一章介绍了计算机辅助设计工具，例如电路图输入和仿真软件。基本存储单元（触发器）的逻辑功能放在第7章讨论，而一些重要的由触发器组合成的MSI和LSI放在第8章，涉及到寄存器、计数器和数据存储器件。第9章论述包括组合逻辑和触发器在内的可编程器件的基本功能。第10章介绍传统的设计过程和使用算法状态机图表作为同步时序逻辑和简单状态机的设计工具。

本书中，关于符号的使用是恰当的，但没有一种符号被广泛接纳。在正向逻辑符号中已经使用了多年的小圆圈，用来表示逻辑反，然而目前在混合逻辑系统中也用半箭头表示低有效信号。因为保留半箭头排它地代表低有效信号减少了二义性，作者在详述2～4章中提出的一般设计过程中使用它，强调了这种符号。然而值得注意的是，目前使用的大多数电路图输入和仿真软件包用的是小圆圈而不是半箭头，不能任意摒弃低有效信号的小圆圈。它的应用在4.3节中讨论，你会发现其例子遍布全书（如在图4-76、图4-79、图5-38中），在实际工作中也可能会遇到这些符号。

在可编程逻辑器件表示接线（或者没有）时这种情况就不再混淆了。从第9章到第11章，“x's”，实心圆圈或实心矩形表明连接，而空心圆圈或无符号表示没有连接。因为读者很可能会遇到以上提到的所有或部分问题，所以一定要对每一个问题和例子给予考虑，以避免在符号意义方面的歧义。

本书既可作为数字系统设计方面入门性课程的教科书，也可作为其他教科书的补充材料。本书具有Schaum大纲系列的特征，包含大量有用的例题和带有答案的补充习题。在设计上，对于一个具体的问题经常有几种答案，注意到这一点是很重要的。在这种情况下，作者在选择给定的方案时使用了最好的判断，并且适时地在连续章节中随着新技术的阐述对于有代表性的一组问题提出了另外的方法。

你会注意到，书中的许多逻辑和时序方框图是由计算机生成的，几类关于电路原理图输入和仿真结果在书中和附录C中给出。这是很自然的结果，事实上如果没有提供电子设计自动化软件(EDA)，解决数字系统的设计就会变得越来越困难。作者所使用的LogicWorks™，比来自Capilano计算机系统有限公司的DesignWorks™数字逻辑设计软件稍小一点的版本，因为它的仿真包用户界面极其友好，并且对于教师和学生来说价格也相当吸引人。有兴趣的读

者可以写信给406－960 Quayside Drive, New Westminster, B. C., Canada V3M 6G2 的 Capilano, 或者打电话(800)444－9064 或(604)522－6200, 以获得更多的信息甚至演示盘。

我们对 Charles Schuler 教授提出的意见表示感谢，他审阅了手稿并且以建设性的鼓励调动我们的热情。我们也对在 RIT 就读的几届本科生表示谢意，他们在习题解答的准确性和教学的有效性上给了我们无价的反馈。和文字相结合的成百张框图的创建应归功于一个或几个辛勤的秘书和/或艺术家（令人难以置信的耐心）。需要指出的是，这项工作是在家中用一台 Apple Macintosh 计算机完成的，它运行两个显示屏幕，一个用来画图，另一个用来录入文字，它使这项工作相当好管理并且充满乐趣。最后，我们确实要特别感谢我们的妻子 Mary Palmer 和 Marjorie Lu Perlman 的耐心和支持，在我们投入这项工作的许多深夜和周末，她们却忍受孤独。

JAMES E. PALMER
DAVID E. PERLMAN

(O-1514.0101)
责任编辑：陈玉琢

SCHAUM'S ouTlines

全球销量超越3000万的
“全美经典学习指导系列”
是您的最佳
学习伴侣！

40年来最畅销的教辅系列
全美著名高校资深教授倾力之作
国内重点高校任课教师全力推荐并担当翻译
省时高效的学习辅导，全面详细的习题解答
迄今为止国内最全面的教辅系列
覆盖大学理工科专业

全美经典学习指导系列

概率和统计	2000工程力学习题精解	电气工程基础
统计学	工程力学	工程电磁场基础
离散数学	3000物理习题精解	数字信号处理
Mathematica使用指南	流体动力学	数字系统导论
数理金融引论	物理学基础	数字原理
机械振动	材料力学	电机与机电学
微分方程	2000离散数学习题精解	基本电路分析
统计学原理（上）	工程热力学	信号与系统
统计学原理（下）	数值分析	微生物学
微积分	量子力学	生物化学
静力学与材料力学	有机化学习题精解	生物学
有限元分析	3000化学习题精解	分子和细胞生物学
传热学	大学化学习题精解	人体解剖与生理学
近代物理学	电路	

http://www.sciencep.com http://www.mheducation.com

ISBN 7-03-009619-3
9 787030 096197 >
ISBN 7-03-009619-3/O · 1514
定 价：36.00 元

目　录

第 1 章　数制和二进制 …… **1**
1.1　引言 …… 1
1.2　数制 …… 1
1.3　不同基数间的转换 …… 3
1.4　基本二进制运算 …… 5
1.5　编码 …… 6
1.6　错误检验和校正 …… 10
第 2 章　组合逻辑 I 的设计 …… **21**
2.1　组合逻辑 …… 21
2.2　真值表 …… 21
2.3　布尔表达式和基本逻辑函数 …… 23
2.4　布尔表达式与真值表的关系 …… 25
2.5　逻辑框图 …… 26
第 3 章　组合逻辑 II 的设计方法 …… **42**
3.1　引言 …… 42
3.2　布尔代数基础 …… 42
3.3　硬件实现 …… 45
3.4　卡诺图基础 …… 45
3.5　卡诺图的应用 …… 48
第 4 章　硬件和混合逻辑规则 …… **69**
4.1　引言 …… 69
4.2　门电路硬件 …… 69
4.3　将混合逻辑作为设计工具 …… 70
4.4　混合逻辑的描述规则 …… 75
4.5　混合逻辑在解决冲突时的使用 …… 76
第 5 章　中规模集成电路和大规模集成电路元器件 …… **100**
5.1　引言 …… 100
5.2　多路转换器 …… 100
5.3　译码器和多路分配器 …… 104
5.4　只读存储器（ROM） …… 104
第 6 章　时序图 …… **127**
6.1　引言 …… 127
6.2　微时序图 …… 127
6.3　冲突 …… 129
6.4　宏时序图 …… 130
6.5　时序仿真 …… 131
6.6　组合电路中的反馈 …… 132
第 7 章　触发器 …… **147**
7.1　引言 …… 147
7.2　基本锁存器 …… 147

7.3　无抖动开关 …… 148
7.4　时钟 RS 触发器 …… 148
7.5　JK 触发器 …… 150
7.6　带有预置端和清零端的 JK 触发器 …… 152
7.7　触发器中的信号的传输 …… 152
7.8　其他触发器类型 …… 154
7.9　触发器的触发和时序 …… 156
7.10　亚稳度 …… 157
第 8 章　触发器的组合 …… **177**
8.1　寄存器 …… 177
8.2　并行-串行转换 …… 179
8.3　波纹计数器 …… 179
8.4　速率乘法器 …… 182
8.5　随机存取存储器（RAM） …… 183
第 9 章　专用器件 …… **203**
9.1　引言 …… 203
9.2　程序设计技术 …… 203
9.3　PROMS 和 EPROMS …… 203
9.4　可编程阵列逻辑（PAL） …… 205
9.5　可编程逻辑阵列（PLA） …… 209
9.6　门阵列 …… 209
9.7　可编程门阵列 …… 211
9.8　全定制设计 …… 211
第 10 章　简单状态机设计 …… **230**
10.1　引言 …… 230
10.2　用 D 触发器设计的传统状态机 …… 230
10.3　用 J-K 触发器设计 …… 233
10.4　可编程逻辑器件的设计 …… 235
10.5　ASM 表 …… 236
10.6　从 ASM 表设计：用最少的 F.F（触发器）实现布尔方程 …… 238
10.7　从 ASM 表设计：单状态（one-hot）控制器的实现 …… 240
10.8　从 ASM 表设计：状态表输入到可编程器件 …… 240
10.9　状态机的时钟 …… 242
10.10　状态机中的初始化与锁定 …… 243
第 11 章　电可编程功能 …… **267**
11.1　引言 …… 267
11.2　基本元件 …… 267
11.3　可编程门阵列 …… 268
11.4　算术逻辑单元 …… 271
11.5　可编程寄存器 …… 274
附录 A　基本布尔定理和恒等式 …… **288**
附录 B　标准逻辑符号 …… **289**
附录 C　数字逻辑模拟的一些说明 …… **294**

第 1 章　数制和二进制

1.1　引言

在现代电子系统中，需要进行电子化存储和处理大量的存在于电子噪声和干扰信号中的数据。数据通常以二进制(二值)的形式存在，因为这样使得可靠并且容易复制的存储和计算器件得以使用。它们集成了大量的逻辑互连电子器件，从而构成了集成电路。这样的器件含有上千个(在很多情况下，往往是数百万个)晶体管。它们本身能够抵抗由干扰引起的错误，因为描述两个二进制状态的电压或电流足以被区分开，从而避免了由寄生干扰引起的错误。各种不同的数据编码和检错方案，例如格雷编码和奇偶校验，经常用来减少没有被检测到的低概率的错误。因此，在数字处理中广泛地使用二进制，理解二进制与其他数制之间的关系、数制的一般特点和从一种数制到另一种数制的转换方法是有意义的。

1.2　数制

在日常使用中，用具有 10 个数码(0,1,2,3,4,5,6,7,8,9)的十进制(以 10 为基数)来表示数。这种数制的计量方法采用位置记数法，其中，分配特定数码的值由数码在数码序列中的位置决定。例如以 10 为基数的数字 853 828。数码 8 在序列中出现了 3 次，但每一次出现有不同的权值，因为数码处于相应的基数指数的不同的位置。排列如下所示：

10^5	10^4	10^3	10^2	10^1	10^0	列权
8	5	3	8	2	8	数码

$$853\,828 = 8\times 100\,000 + 5\times 10\,000 + 3\times 1000 + 8\times 100 + 2\times 10 + 8\times 1$$

最左边的 8 以 10^5 加权，接下来的 8 以 10^2 加权，最后一个 8 以 10^0 加权。这种位置记数法很容易地扩展到小数部分，这种情况下使用基数 10 的负次幂：

$$0.725 - 7\times 10^{-1} + 2\times 10^{-2} + 5\times 10^{-3}$$

二进制

用任何基数来表示数字是可能的。在二进制的情况下，基数是 2 仅需要两个符号(0 和 1)，每个数码为 1“位”，并且再次采用位置记数法。只要写出每一个 2 的幂的十进制当量，分别乘以正确的二进制数码，然后将这些结果相加就可以得到任何二进制数的十进制当量。

例 1.1　将二进制数 1100111.1100 表示为十进制数(以 10 为基数)。

因为整数部分有 7 个数码(位)，最高位的权值为 2^6 即 64。它对应的十进制数可很容易地按下面方法来计算：

$$\begin{aligned} 1100111 &= 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0 \\ &= 1\times 64 + 1\times 32 + 0\times 16 + 0\times 8 + 1\times 4 + 1\times 2 + 1\times 1 \\ &= 103_{10} \end{aligned}$$

对于小数部分，

$$\begin{aligned} .1101 &= 1\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4} \\ &= 1\times 0.5 + 1\times 0.25 + 0\times 0.125 + 1\times 0.0625 \\ &= 0.8125_{10} \end{aligned}$$

二进制仅需要两个符号，这在理论上适合由电子器件来表示它们，因为这仅需要两个容易区分的状态，例如关和断(导通和截止)。

考虑到在数字处理系统的实现中噪声和干扰的影响，二进制的优点就得以很好的展示。

在二进制情况下，当从存储器传输和发送数据的时候，接收器必须要确定给定信号接近于哪一种电平。确定的阈值可设置在两个电平的中间，因此任何附带的小于信号电平和阈值间差值的噪声就被忽略掉了。另一方面对于十进制的存储器，因为必须将给定的范围分成 10 个分开的等级，具有同样的电压范围但分配给信号的电压不同而使系统的抗扰度小得多(见图 1-1)。

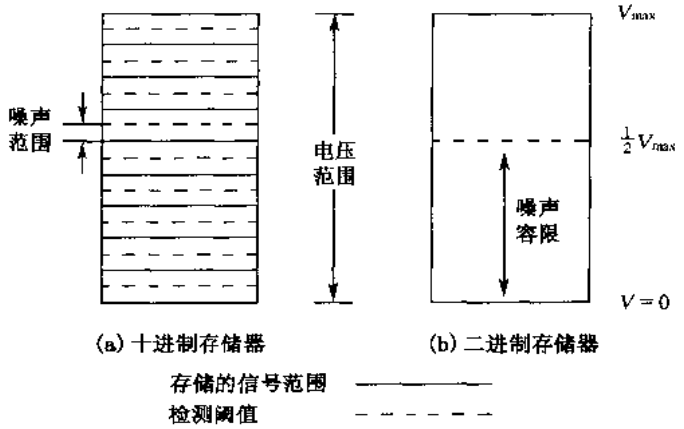


图 1-1

数字系统中，我们提到的称作噪声容限的数字指标，其定义为没有引起不期望的输出变化所能忍受的最大电压(或电流)。

例 1.2 比较具有理想元器件的二进制和十进制系统的基本噪声容限。

对于二进制的情况，将 1 存为 V_{max}[典型值为 5 伏(V)]，将 0 存为大约 0 伏。阈值将被设为 $V_{max}/2$，任何小于该值的噪声将被忽略。十进制系统中，在 0 和 V_{max}之间将有 10 个相等的存储空间等级(0，$V_{max}/9$，$2V_{max}/9$，等等)并且将在相临的存储等级的中间设置阈值($V_{max}/18$，$3V_{max}/18$，等等)。任何大于 $V_{max}/18$ 的噪声会导致读数据错误。对于 $V_{max}=5V$ 的二进制系统，噪声容限为 $5/2=2.5V$。另一方面具有同样 V_{max}的十进制系统的噪声容限仅仅是 $(5/9)/2=0.28V$，与二进制系统相比，这明显不合乎需要。

八进制和十六进制

因为二进制仅用两个符号，所以对于数字系统中数据存储和处理有很大的实用价值，但是用二进制表示给定的数，比相应的十进制数由更长的位序列组成，如果要人工输入数据，仅需要一个 2 键的键盘，但这些键将被敲击多次。经常采用成组处理二进制数来解决数据输入问题。

八进制数根据下表使用 3 位一组的数。

二进制	八进制位
000	0
001	1
010	2
011	3
100	4
100	5
110	6
111	7

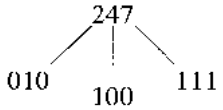
每一个八进制符号代表与 3 位一组的二进制数相同的数字，这 8 个符号组成了基数为 8

的数制。这种情况下数据输入需要一个 8 键的键盘，但敲击次数仅是二进制键盘的 1/3。

例 1.3　将八进制数 247 表示为十进制数和二进制数。

八进制数采用位置记数法，最低位（最右边）以 $8^0=1$ 加权，最高位以 $8^2=64$ 加权。因此 $247=2\times64+4\times8+7\times1=167_{10}$。

参照上表，可见通过分组很容易就可以得到八进制到二进制的转换：



通过确定与所得的二进制数 10100111 相等的十进制数可以很容易地证明转换的正确性。注意：最前面的 0 可以去掉。最高位在左边的第 8 的位置上，所以以 $2^7=128$ 来加权。因此，$10100111=128+32+4+2+1=167$。

不同基数的数，它们的通用转换方法将在 1.3 节中讨论。

十六进制记数法将成组的思想扩充到 4 位，组成了以 16 为基数的数制。位组与相应的十六进制符号如下表所示。

二进制	十六进制	二进制	十六进制	二进制	十六进制	二进制	十六进制
0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

十六进制的符号 0 到 9 与最前面的 10 个 4 位二进制数组的十进制数的值是相等的。因为没有单个十进制数码能表示比 9 大的数，为表示最后 6 组我们需要新符号。如上表所示，用字母表中前 6 个字母实现这一目的。十六进制数需要 16 个键的键盘，但敲击率仅是二进制键盘的 1/4。

例 1.4　十六进制-二进制转换。(a) 将 1101011100110 转换为相应的十六进制数。(b) 将 4B2F 转换为二进制数。

(a) 11010111001100 = (0001)(1010)(1110)(0110)　以 4 分组
= 1 A E 6　分别转换
= $1AE6　通常用美元符指示十六进制数

(b) 4B2F = (0100)(1011)(0010)(1111)　分别转换
= 0100101100101111　取消分组
= 100101100101111　去掉最前边的 0

1.3　不同基数间的转换

以下介绍的是一种通用方法，用于任何基数之间的转换。

1. 整数和小数部分分开转换。

2. 整数部分用新基数连续除来转换，用产生的余数序列创建一个新数。根据旧基数实现算法。

3. 小数部分用新基数连续乘来转换，被转换的小数部分用所产生的整数来表示。再次根据旧的基数实现算法。

例 1.5　将十进制数 278.632 转换为相应的二进制数。

第 1 步：整数是 278。小数是 0.632。

第 2 步：整数转换

除	所产生的余数	
2)278		
2)139	0	
2)69	1	
2)34	1	
2)17	0	向上读形成：100010110
2)8	1	
2)4	0	
2)2	0	
2)1	0	
0	1	最高位(MSB)

提示：一旦余数已经形成，在算法中它将不再起任何作用。整数处理就终止。

第 3 步：小数转换

乘	所产生的整数	
0.632×2=1.264	1	MSB
0.264×2=0.528	0	
0.528×2=1.056	1	
0.056×2=0.112	0	
0.112×2=0.224	0	
0.224×2=0.448	0	向下读形成：.101000011
0.448×2=0.896	0	
0.896×2=1.792	1	
0.792×2=1.584	1	

提示：一旦整数形成它将不再起任何作用，这个过程可能不会终止，通常执行到精度满足要求为止。

例 1.6 将十进制数 123.456 转换为相应的八进制数(基数为 8)。

整数转换：

除	所产生的余数	
8)123		
8)15	3	
8)1	7	向上读形成 173
0	1	

小数转换：

乘	所产生的整数	
$0.456\times8=3.648$	3	
$0.648\times8=5.184$	5	向下读形成 0.3513
$0.184\times8=1.472$	1	
$0.472\times8=3.776$	3	

过程被任意地终止。

$$123.456_{10}=173.3513 \quad (\text{大约})$$

检查：

$$173_8=1\times64+7\times8+3\times1=123_{10}$$

$$0.3513_8=3\times0.1250+5\times0.0156+1\times0.0020+3\times0.0002=0.4556_{10}$$

1.4　基本二进制运算

以上讨论的所有数制都是位置加权，这使得每次在一位上做带进位的运算成为可能。通过对单个位的规则重复应用能够逐步获得全加和倍乘表。

二进制加法

二进制加法表非常简单，如下所示，涉及到的两个数字用 X 和 Y 表示。C_i 是低位加法器向前的进位。　　这是典型的 1 + 1 = 2

进位，C_i	0	0	0	0	1	1	1	1
数字 X	0	0	1	1	0	0	1	1
数字 Y	0	1	0	1	0	1	0	1
	0	1	1	10	1	10	10	11

提示：所有的单独位的加法中在结果超过 1 的地方就会出现进位输出位。

例 1.7　两个长位串的相加。

相加时，每一对数按位相加会产生一个结果，并且，当结果超过 1 的时候会产生一个进位输出位。如下图所示，这个进位输出位会成为下一个高阶位的进位。例如，当 X = 1，Y = 1，进位也是 1 时，结果为 3(二进制为 11)，最左边的位的十进制的值是 2，就被进位到了高阶的一列，在正下方的结果的位置留下 1。

	011110001
数字 X	1010111001
数字 Y	0011010101
结果	1110001110
进位输出位	0011110001



二进制减法

采用借位和得到差值，以类似的形式进行减法的论述。然而实际上，减法是由与做加法相同的硬件电路通过补码运算来完成的。在二进制情况下，负数由相应的正的二进制数的 2 的补码来表示。将 X 转换成 −X 从而获得 X 的 2 的补码然后加上 Y，来实现从另一个数 Y 中减去一个给定的数 X。在这种方法中，将最左边的位指定为符号位(0 代表正，1 代表负)，除忽略了由符号位相加产生的进位输出位，将其看作和其他位一样。

通过交换原二进制数的 0 和 1，将这个结果再加 1 可以获得二进制数的 2 的补码。

例 1.8 从 230_{10}中减去 185_{10}，将其转换成二进制，采用 2 的补码算法。

用做计算所需要的二进制位的数目由最大的数决定(包括结果)，这种情况下，数字 230 是最大的，需要 8 位再加上附加的符号位。如此，可将 185 的相应的二进制数写为 010111001。

注意：最前边的 0 对数值是没有影响的。

我们通过取它的 2 的补码将其转换为负数。

第 1 步：对 0 和 1 取反

$$010111001 \longrightarrow 101000110$$

第 2 步：加 1

$$\begin{array}{r} 101000110 \\ + \qquad\qquad 1 \\ \hline 101000111 = -185_{10} \end{array}$$

然后再用 9 个数位，将 230 转化为二进制数，并且加上上面第 2 步的结果：

$$\begin{array}{rl} +230 = & 011100110 \\ -185 = & \underline{101000111} \\ & 1000101101 \end{array}$$

忽略进位的符号位(左边的附加位)得到了 000101101，它最左边的位是 0，表明结果是正的。

检查：将 000101101 转换为十进制为 +45。

例 1.9 从 185_{10}中减去 230_{10}，将其转换成二进制，使用 2 的补码算法。

230 相应的二进制数是 011100110，对 0 和 1 取反再加 1 获得它的二进制补码。

$$-230 = 100011010$$

接着，将这个值加上 185 的相应的二进制数：

$$\begin{array}{r} -230 = 100011010 \\ +185 = 010111001 \\ \hline 111010011 \end{array}$$

最左边的位是 1 表明结果是负数。为得到所要求的值，因为 $-(-X)=X$，我们取结果的补码形式。

$$\begin{array}{r} 000101100 \\ + \qquad\quad 1 \\ \hline 000101101 \end{array}$$

该数对应的十进制数是 45，我们早已确定它是负数。

1.5 编码

二-十进制编码

二-十进制编码的本质是以二值形式(二进制)对十进制数进行编码。十进制数的每一位按顺序由它相应的 4 位二进制数来表示。4 位中最小的数代表十进制整数 0。因为 4 位二进制数有 16 种可能的组合，在 BCD 编码中有 6 个没有用。

例 1.10 比较十进制数 278 的二进制和 BCD 码描述。

在例 1.5 中说明 278 的相应的二进制数是 100010110。通过将给定的十进制数作为一个

整体来处理完成转变。在 BCD 码转换中，对每一个十进制数的位分别进行编码：

$$278_{10} = (0010)(0111)(1000) = 001001111000 \quad (BCD)$$
$$\qquad\quad\ \ 2 \quad\ 7 \quad\ 8$$

格雷码

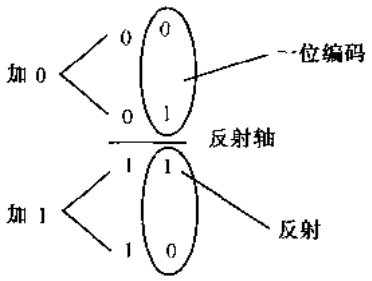
另外一种在工程上比较重要的二值编码是格雷码，有时称做反射二进制编码。它不是位置加权编码，并且因为这个原因它不适合做算术运算。

1 位格雷码与 1 位二进制编码相同：

0

1

2 位格雷码通过在虚构的镜子中反射 1 位格雷码得到，如下图所示。对于第二位，把 0 加到反射轴的上方，1 加到反射轴的下方。



以 2 位的格雷码为基础进行反射，然后再在反射轴的上方加 0，反射轴的下方加 1 从而形成了 3 位的格雷码。

000
001
011
010
110　　反射轴
111
101
100

这个过程可以重复位数为任意的数字。

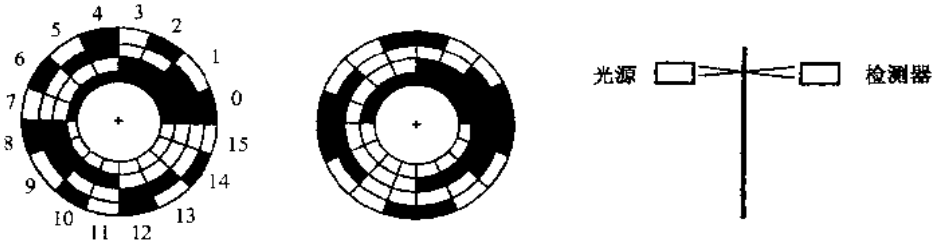
在格雷码的产生过程中所用的反射会保证格雷码有单位距离的特性，这意味着连续的码组之间仅有一位不同。例如，表 1.1 所示的 4 位的格雷码及其相应的十进制和二进制数。随着数字在序列中的前进而观察格雷码的单位距离，将这种特性与二进制码相比较。例如，从 7 到 8，在二进制中所有的位都改变了。

表 1.1

十进制数	二进制数	格雷码	十进制数	二进制数	格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

格雷码的主要应用是在测量方面，其中的典型例子在例 1.11 中说明。

例 1.11 位置编码盘。在机器人系统中，“手臂”的运动常常由发出某种使关节旋转命令的信号的微处理器来指挥。命令常应用在能控制电机方向和速度的电路中。计算机为了保证执行所要求的命令，必须要知道关节的实际位置并且将它与命令比较。位置测量的实现经常是通过将一个小编码盘连在电机的轴上；编码盘由同心轨道组成。如图 1-2 所示，轨道具有透明和不透明扇区的确定形式。每个轨道分别与一个光源和一个光检测器相联系。当轨道的透明扇区位于光源和检测器之间时，光被发送出去，检测器产生了电信号。当轨道的不透明扇区位于光源和检测器之间时不会产生任何输出信号。



(a) 二进制编码　　(b) 格雷编码　　(c) 编码盘（侧视）

图 1-2

图 1-2(a)中的编码盘为二进制编码，最外层的轨道为相应的最低位。分开的光源-检测器呈辐射状排列在每个轨道上，当盘旋转的时候我们会看到，经过检测头阵列的每一个扇区会产生惟一的输出组合，可以将其翻译成二进制数。例如，光源-检测器阵列位于盘的上方的垂直线的时候，就认为图示的检测器会产生二进制数 1。如果盘反向旋转超过 1 个扇区时，检测器的输出会表示二进制数 0010。在图 1-2(b)所示编码盘的情况下，扇区由格雷编码识别。

考虑二进制编码的情况，盘所处的位置使检测器位于 7 和 8 的分界线上。注意：对于所有的轨道界线，两边的扇区是不同的。光源和检测器的排列有一定的偏差，光源的发射也不是零宽度；光线会传播。如果光源正在线上，它可能会引起检测器输出。因此产生的二进制数可能是从全 0 到全 1 间的任何值，这依赖于光源和检测器的排列和光的传播。过失误差就可能会出现。

另一方面，如果我们使用格雷编码盘，因为单位距离的特性，每个扇区仅仅有一个轨道在该扇区的界线两侧发射的光是不同的，因此仅有一个格雷编码位可能发生错误。格雷编码方案仅根据临近的扇区产生数字，所以不会产生大的误差。

虽然格雷码非常适合测量，但它不用于运算，因为如上面所提到的，它不具有位置加权的特点。

二进制码和格雷码间的转换

注意：格雷码可以认为是相应二进制码的“差分”形式，就可以有效地处理这个问题。

A. 从二进制码到格雷码的转换

1. 在两种数制中最左边的位相同。

2. 从左向右读二进制数。在格雷码中有变化(0 到 1 或 1 到 0)就产生 1，否则产生 0。

例 1.12 二进制码-格雷码转换。将二进制码 01101001101 转换成格雷码。

二进制码：　0 1 1 0 1 0 0 1 1 0 1

D C S C C C S C S C C　　S = 相同；C = 不同；D = 照抄

格雷码：　0 1 0 1 1 1 0 1 0 1 1

01101001101(二进制码) = 01011101011(格雷码)

B. 从格雷码到二进制码的转换

1. 在两种数制中最左边的位相同。

2. 从左向右读格雷码数。1 表示下一个二进制数位必须改变，0 表示下一个二进制数位与左边的数位相同。

例 1.13　格雷码-二进制转换。将格雷码数 1000110101010 转换成相应的二进制数。

格雷码：　1 0 0 0 1 1 0 1 0 1 0 1 0

D S S S C C S C S C S C S

二进制码　1 1 1 1 0 1 1 0 0 1 1 0 0

01101001101(二进制码) = 01011101011(格雷码)

ASCII 码

计算机存储和处理的数据并不都是数字。因为二进制的使用价值，其他数据类型也以二值(与二进制一样)形式保存。最常遇到的字母码是美国标准信息交换码，表 1.2 中包含一些典型的键盘字符；这个列表是不全面的，仅仅是为了演示。提示：十进制数作为 BCD 编码数列出，由 011 起头(9 = 011 1001)，其他的 3 位前缀代表非数字数据。

表 1.2　部分 ASCII 码字符集

键　值	ASCII 码	十六进制	键　值	ASCII 码	十六进制
A	100 0001	41	U	101 0101	55
B	100 0010	42	V	101 0110	56
C	100 0011	43	W	101 0111	57
D	100 0100	44	X	101 1000	58
E	100 0101	45	Y	101 1001	59
F	100 0110	46	Z	101 1010	5A
G	100 0111	47	空格	010 0000	20
H	100 1000	48	(	010 1000	28
I	100 1001	49	)	010 1001	29
J	100 1010	4A	+	010 1011	2B
K	100 1011	4B	0	011 0000	30
L	100 1100	4C	1	011 0001	31
M	100 1101	4D	2	011 0010	32
N	100 1110	4E	3	011 0011	33
O	100 1111	4F	4	011 0100	34
P	101 0000	50	5	011 0101	35
Q	101 0001	51	6	011 0110	36
R	101 0010	52	7	011 0111	37
S	101 0011	53	8	011 1000	38
T	101 0100	54	9	011 1001	39

1.6 错误检验和校正

奇偶校验

在电子数据处理过程中，消除随机噪声是不可能的，伴随而来的结果是可能发生错误。处理这种情况最常用的方法是加入称为奇偶校验位的冗余位。每一个奇偶校验位都与一组数据位相联系并且在这组数位中选择 1 的总数来进行奇校验或偶校验。其后在任何情况下，能够计算整组数中 1 的个数，并且将其与最初创建的奇偶校验位进行比较，所谓的简单奇偶校验位总会检测出一组数中单个数字的错误，在单个错误普遍发生的环境中这种方法是非常有效的。例如，考虑以奇校验进行编码的 4 位 BCD 字符，奇偶校验位是最右边的位(参照表 1.3)。

表 1.3

十进制数	BCD 码	添加的奇偶校验位
0	0000	00001
1	0001	00010
2	0010	00100
3	0011	00111
4	0100	01000
5	0101	01011
6	0110	01101
7	0111	01110
8	1000	10000
9	1001	10011

如果要处理 5(BCD 码为 0101)，数位组 01011 作为奇偶校验 BCD 码将被存储。如果后来这个数位组被取出来时成了 11011(第 1 位发生错误)，现在的奇偶校验是偶数，错误将被检查出来。

纠错

奇偶校验也可以用在能够检查和改正某种错误类型的特殊组合中。一个例子就是图 1-3 所示的所谓的汉明码(Hamming code)，在这种情况下，用 4 个数据位(D_1, D_2, D_3, D_4)创建附加的 3 个奇偶校验位(P_1, P_2, P_3)然后将它们添加到最初的 4 个数据位，形成了 7 个可以成组存储和发送的数据位。

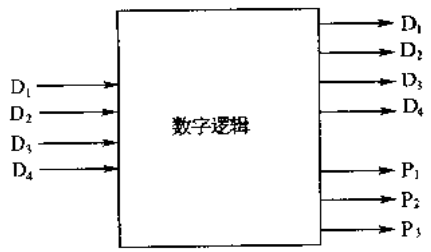


图 1-3

汉明码的实现能够用表 1.4 中所定义的逻辑来完成。从 3 个数据位的不同组合中形成 3 个校验组，并且对每个组产生一个奇偶校验位。校验组的成员在相应的位列中由 X 指出。可

以注意到，7 个数位中的每一位在校验组中有惟一的组合。如果假设由于噪声或干扰仅有一个位发生错误，那么惟一的一种组合就有同样位置错误的结构产生，对应一位可能发生的错误。对奇偶校验错误形式的了解可能识别发生错误的位。因为这是二进制，知道哪一位发生错误就有可能将其改正。

表 1.4　汉明码的产生

	D_4	D_3	D_2	P_3	D_1	P_2	P_1
校验组 1#	X		X		X		X
校验组 2#	X	X			X	X	
校验组 3#	X	X	X	X			

考虑数据 D_4, D_3, D_2, D_1 是 1011 的情况。参照表 1.4，对于校验组 2 我们可以看到，在奇校验要求下，因为 D_1、D_4 是组中的成员，D_3 是 0，D_2（非成员）不用考虑，所以 P_2 为 1。将奇校验产生规则与表 1.4 定义的算法相结合就会产生校验位 $P_3, P_2, P_1 = 110$。汉明码以 $D_4, D_3, D_2, P_3, D_1, P_2, P_1$ 的顺序发送，因此在提出的例子中，逻辑电路的输出会以 1011110 形式发送。

假设现在左边数第二个数据位是错误的；例如收到的序列是 1111110。收到数据后，组将分别被校验，并且参照表 1.4 表明校验组 2 和 3 发生错误（表明为偶校验）。组 2 和组 3 发生错误而组 1 成功发送的惟一情况是当 D_3 发生错误。它的错误值是 1，因此应当改为 0。如果只有 1 个校验位发生错误，那么仅仅有 1 个附加在这个位的校验组发生错误，因此将其识别出来加以改正。

非奇偶检错码

采用非奇偶检验码得到的其他检错技术也是存在的。例如，在一些跨洋无线电通信系统中，进行编码发送，因此如果 1 个字符通信错误，接收器的错误检测电路会自动产生再传输请求（ARQ）。所用的编码取出 1 个（可能是 70 个中的 1 个）数据字符，将它配置成 8 位具有确切的 4 个 1 和 4 个 0。在接收器端，计算字符中 1 的个数，如果结果不是 4，就认为产生了错误。编码会检测所有的奇数个错误，也会检测所有的偶数个错误，该处 1 到 0 的改变没有通过 0 到 1 的改变来实现。

习 题 解 答

1.1　把十进制数 234.567 转换为相应的二进制数。进行 9 步转换确定小数的精度。

解

整数转换	余数	小数转换	整数
2)234		0.567 × 2 = 1.134	1
2)117	0	0.134 × 2 = 0.268	0
2)58	1	0.268 × 2 = 0.536	0
2)29	0	0.536 × 2 = 1.072	1
2)14	1	0.072 × 2 = 0.144	0
2)7	0	0.144 × 2 = 0.288	0
2)3	1	0.288 × 2 = 0.576	0
2)1	1	0.576 × 2 = 1.152	1
0	1	0.152 × 2 = 0.304	0

二进制数是 1101010.100100010，最右边系数的权值是 2^{-9}，因此任何由于截断产生的误差会小于 2^{-9}或 0.001953125。

1.2 把 345.678(十进制数)转换为相应的基数为 6 的数，转换误差小于 0.001。

解 因为 $6^{-3}=0.0046$ 而 $6^{-4}=0.000771605$，所以需要四个小数位

整数转换	余数	小数转换	整数
6)345		0.678×6=4.068	4
6)57	3	0.068×6=0.408	0
6)9	3	0.408×6=2.448	2
6)1	3	0.448×6=2.688	2
0	1		

345.678(十进制数)－1333.4022(基数为 6)

1.3 把 217(八进制数)转换成二进制数。

解 使用通用转换规则。把要转换的数(217)用新的基数(2)来除。以原来的基数 8 实现算法。以基数 8 除的例子：

```
    1 0 7
2 ) 2 1 7
    2
    ---
      1 7   <-------- 15(10)
      1 6   <-------- 以 10 为基数是 2 × 7 = 14，以 8 为基数是 16：
      ---
        1   余数  1×8^1 + 6×8^0
```

```
      4 3
    -----
2 ) 1 0 7
    1 0     <-------- 8(10)
    -----
      0 7
      0 6
    -----
        1
```

完成转换，注意没有小数部分。

	余数
2)217	
2)107	1
2)43	1
2)21	1
2)10	1
2)4	0
2)2	0
2)1	0
0	1

217(八进制)＝10001111(二进制)

1.4　将十六进制数 B73D 表示为相应的八进制数。

解　☞ 将十六进制数转换成二进制数(4 位一组)。

$$\begin{aligned}\$\,B73D &= (1011)(0111)(0011)(1101)\\ &= 1011011100111101(二进制)\end{aligned}$$

将二进制数转换成八进制数(3 位一组)。从右边开始必要时在前边添加 0。

$$\begin{aligned}1011011100111101(二进制) &= (001)(011)(011)(100)(111)(101)\\ &= 1\quad 3\quad 3\quad 4\quad 7\quad 5\\ &= 133475(八进制)\end{aligned}$$

1.5　将八进制数 23 456 表示为十进制数和 BCD 码数。

解　☞

$$\begin{aligned}23\,456(八进制) &= 2\times 8^4 + 3\times 8^3 + 4\times 8^2 + 5\times 8^1 + 6\times 8^0\\ &= 2\times 4096 + 3\times 512 + 4\times 64 + 5\times 8 + 6\times 1\\ &= 8192 + 1536 + 256 + 40 + 6\\ &= 10\,030\end{aligned}$$

分别将十进制的数转换为二进制。

$$\begin{aligned}10\,030(十进制) &= (0001)(0000)(0000)(0011)(0000)\\ &= 0001\ 0000\ 0000\ 0011\ 0000\ (BCD)\end{aligned}$$

1.6　将 BCD 数 0010 0111 1001 0101 转换为十六进制数。

解　☞ 首先,将 BCD 数转换为十进制数:

$$(0010)(0111)(1001)(0101) = 2795$$

第 2 步,将十进制数转换为二进制数。

2)2795	
2)1397	余数 = 1
2)698	余数 = 1
2)349	余数 = 0
2)174	余数 = 1
2)87	余数 = 0
2)43	余数 = 1
2)21	余数 = 1
2)10	余数 = 1
2)5	余数 = 0
2)2	余数 = 1
2)1	余数 = 0
0	余数 = 1

向上读产生 101011101011(二进制数)。

第 3 步,以 4 位分组形成十六进制数码。

$$(1010)(1110)(1011) = AEB$$

1.7　将 F5.3B(十六进制)转换为相应的八进制数。

解　☞ 为方便起见,用间接地转换为二进制的方法。

$$\begin{aligned}\$\,F5.3B &= (1111)(0101).(0011)(1011)\\ &= 11110101.00111011\end{aligned}$$

相对于二进制小数点以 3 位分组,转换成八进制。

$$11110101.00111011 = (011)(110)(101).(001)(110)(110) = 365.166(八进制)$$

1.8 把 $A47 和 $854 相加。以十六进制形式做运算，提示：十六进制码是加权码。

解 ☞ 逐位相加：

$$\begin{array}{r} A47 \\ +\,854 \\ \hline 129B \end{array}$$

7 + 4 = 11（十进制）= B（十六进制）

4 + 5 = 9（十进制）= 9（十六进制）

A + 8 = 18（十进制）= 2 加上一个进位（十六进制）

1.9 把 234（十进制）加到 189（十进制）上并且以二进制的形式做运算。

解 ☞ 两个数都是正数；过程是简单相加。首先将其转换为二进制数。

234（十进制）= 0011101010（二进制）

189（十进制）= 0010111101（二进制）

用 1.4 节说明的方法相加。

进位输入	011111000
234	0011101010
189	0010111101
求和	0110100111
进位输出	0011111000

和为正数（符号位 = 0）并且可将其转换成 +423（十进制），结果证明无误。

1.10 补码相加可用来完成任何基数的减法，包括以 10 为基数。用 9 的补码相加，从 1462_{10} 中减去 1937_{10}。

解 ☞ 在每个数前加 0 作为符号位。对一个数求补通过从这个数中减去 $B^N - 1$ 的方法，在这里基数为 B，N 是数字的位数。在现在的情况下，B = 10，N = 4，并且要从 9999 中减去第 2 个数（1462）。

$$\begin{array}{rl} 9999 & \\ -\,1462 & \\ \hline 98537 & \leftarrow \text{永远不会有借位，因为所有的数都小于 9} \\ +\,1 & \\ \hline 98538 & \leftarrow \text{1462 的 9 的补码} \end{array}$$

最左边的数位是 9，表明这个数是负数。1462 的 9 的补码与 −1462 相等，可以表明，再次取 9 的补码能够得到原来的数。显而易见这个过程相当于 X = −(−X)。

$$\begin{array}{rl} 99\,999 & \\ -\,98\,538 & \\ \hline 01\,461 & \leftarrow \text{符号位是 0，表明是一个正的结果。} \\ +\,1 & \\ \hline 1462 & \end{array}$$

现在进行题目所要求的减法。

$$\begin{array}{r} 01\,937 \\ +\,93\,538 \\ \hline 100\,475 \end{array}$$

包括附加的符号位进位，我们得到了 475，因为最左边的位是 0，说明它是正数。

1.11 从 176 中减去 204。用 2 的补码以二进制形式求解。

解 ☞ 这些数中的最大数要求 8 位二进制数，因此，带上符号位，N = 9。

从一个 N 位的数中减去由 N 个 1 组成的数可以得到这个数的二进制数的补码。在二进制的情况下，通过简单地对 1 和 0 求反可以很容易地获得。

176 的补码：　010110000→101001111　对所有位求反

加 1：　000000001

2 的补码是：　101010000　与 −176 等值

把 −176 加到 +204 上：　204→ 011001100

−176→ 101010000

1000011100

忽略来自符号位求和的进位输出

和：000011100

符号位是 0，因此这个数是正的。直接转换结果是 +28(十进制)，结果证明无误。

1.12　把 −176(十进制)加到 −204(十进制)上。用 2 的补码以二进制形式求解。

解　结果的值是 380_{10}，需要 9 个二进制数位。包括 1 个符号位，为进行计算我们至少需要 10 个位码。用 2 的补码形式表达这个负数：

$$-176_{10} = 1101010000$$

$$-204_{10} = 1100110100$$

相加：

$$\begin{array}{r} 1101010000 \\ 1100110100 \\ \hline 11010000100 \end{array}$$

忽略最左边的进位输出并且注意到接下来的位(和的符号位)是 1，表明是 1 个负的结果。为求解负数的值，取它的 2 的补码：

$$\begin{array}{rl} 1010000100 \rightarrow & 0101111011 \\ + & \underline{\qquad\qquad 1} \\ & 0101111100 = 380_{10} \end{array}$$

十进制的结果是 −380。

1.13　用 2 的补码运算以二进制形式从 176 中减去 204。

解　因为结果的值小于所给数中的任意一个，N = 9 位。将要减的数取 2 的补码形式(相当于对它取反)。

$$-204_{10} \rightarrow 100110100$$

$$+176 \rightarrow 010110000$$

相加：

$$\begin{array}{rl} -204 \rightarrow & \underline{100110100} \\ & 111100100 \end{array}$$

没有要忽略的进位；忽略符号位。取 2 的补码形式确定大小。

$$\begin{array}{rl} 111100100 \rightarrow & 000011011 \\ + & \underline{\qquad\qquad 1} \\ & 000011100 = 28_{10} \end{array}$$

十进制的结果是 −28。

1.14　从 173(八进制)中减去 356(八进制)。用 8 的补码相加完成。

解　加 1 个最左边(最高阶)的位作为符号位；负数加 7，正数加 0。从 7777 中减去所选的八进制数，取它的补码。

$$\begin{array}{rl} & 7777 \\ - & \underline{0365} \\ & 7412 \\ + & \underline{\quad 1} \\ & 7413 \quad \text{(365 的 8 的补码)} \end{array}$$

相加：

$$\begin{array}{rl} & 0173 \\ + & \underline{7413} \\ & 7606 \quad \text{(以八进制形式相加)} \end{array}$$

八进制加法实现如下：

3 + 3 = 6
7 + 1 = 8（十进制）
= 10（八进制）　　（1 是向左边的进位列）
4 + 1 + 1(进位) = 6
7 + 0 = 7

没有要忽略的进位输出符号位。

结果的说明：最左边的位是 7；因此结果是负的。7606 的 8 的补码 = 0171 + 1 = 0172；因此值是 172。八进制的结果是 7606 或 - 172。

1.15 求格雷码数 1101101001101O 的相应的二进制数。

解 ☞ 1. 最左边的位相同。

2. 从左到右读格雷码数，并且回忆当从左到右进行时，在等值的二进制数中的 1 所表明位码的变化。

格雷码　1 1 0 1 1 0 1 0 0 1 1 0 1 0
　　S＝相同
　　C S C C S C S S C C S C S
　　C＝变化
二进制数　1 0 0 1 0 0 1 1 1 0 1 1 0 0

1.16 给出十六进制数 3A7 对应的格雷码数。

解 ☞ 首先，将十六进制转换为二进制。

$3A7 = (0011)(1010)(0111) = 1110100111　（二进制）

将二进制转换成格雷码。

二进制数　1 1 1 0 1 0 0 1 1 1
　　S＝相同
　　S S C C C S C S S
　　C＝变化
格雷码　1 0 0 1 1 1 0 1 0 0

1.17 求出八进制数 527 对应的格雷码数。

解 ☞ 527(八进制) = (101)(010)(111) = 101010111(二进制) = 111111100(格雷码)

1.18 当在磁道上存储数据块时，有时在行和列上都计算奇偶校验位。对以下所示的数据块求出行和列的奇偶校验位，使用奇校验。

数据
10110
10001
10101
00010
11000
00000
11010

解 ☞ 在行奇偶校验位上，最高位的 0 表明垂直地向下读最左边的列有奇数个 1。下一个位是 1，告诉我们与它相关的数据列有偶数个 1，等等。

数据	带奇偶校验位的数据
	↓列奇偶校验位
10110	101100
10001	100011
10101	101010
00010	000100
11000	110001
00000	000001
11010	110100
	011100←行奇偶校验位

1.19　将数据 D_4, D_3, D_2, D_1 编码成 7 位的汉明码。使用奇校验 P_3, P_2, P_1。

解　参照表 1.4。校验组 1 使用 D_4, D_2, D_1 和 P_1。成员数据是 101 和为 1 的 P_1。校验组 2 使用 D_4, D_3, D_1 和 P_2。成员数据是 101 和为 1 的 P_2。校验组 3 使用 D_4, D_3, D_2 和 P_3。成员数据是 100 和为 1 的 P_3。最后的序列是 $D_4, D_3, D_2, P_3, D_1, P_2, P_1 = 1000111$。

1.20　假设在题 1.19 中编码的序列被接收为 1010111。哪些位是错误的?

解　校验组 1 使用 D_4, D_2, D_1 和 P_1→ 1111（偶）检查有误

校验组 2 使用 D_4, D_3, D_1 和 P_2→ 1011（奇）检查通过

校验组 3 使用 D_4, D_3, D_2 和 P_3→ 1010（偶）检查有误

参考校验组成员图，仅仅以为是组 1 和组 3 的成员但不是组 2 的成员的位是 D_2，因此 D_2 是错误的位。

1.21　如表 1.4 中所定义的，我们已经知道，将数据的 4 位编码成单错校验码(奇校验)。假设仅仅发生了单错，收到的序列 $D_4, D_3, D_2, P_3, D_1, P_2, P_1 = 1011111$，那么正确的数据是什么?

解

校验组 1 = 1111	有误
校验组 2 = 1011	通过
校验组 3 = 1011	通过

仅仅出现在组 1 中的惟一的 1 位数是 P_1，P_1 是错误的，数据位都是正确的。

注意：仅仅一个校验组错误意味着错误发生在奇偶校验位上。

1.22　说明以下的 ASCII 编码序列：

1000001 1000100 1000100 0100000 0110011 0110100 0111001

解　使用表 1.2：

1000001 ⟶A
1000100 ⟶D
1000100 ⟶D
0100000 ⟶空格
0110011 ⟶3
0110100 ⟶4
0111001 ⟶9

结果是"ADD349"。

1.23　比较十进制存储系统和假设的三态系统的噪声容限。这两种系统都利用晶体管中的电流作为存储变量。

解　对于这两个系统假设物理状态在零电流和全电流(I_{max})之间是同样的距离。正如 1.2 节所说明的一样，对十进制系统信号级别和阈值之间的距离是 $I_{max}/18$。对于三态系统信号级别会是 0，$I_{max}/2$ 和 I_{max}。在信号的中间(在 $I_{max}/4$ 和 $3I_{max}/4$ 之间)设定阈值，来自信号的级别的阈值的分割点

是 $I_{max}/4$。因此，三态系统具有大于十进制系统 4.5 倍的噪声容限 $(I_{max}/4)/(I_{max}/18)$。

1.24 为 11 位的数据组设计单纠错编码。

解 对于被编码的组的每一个位必须有校验组成员的惟一组合，因此至少需要 4 个校验组。整个位数是 11 加 4，即 15 位。因为可以将 4 个校验组安排在 16 个不同的组合中，4 个校验组就足够了，并且每个位都能被"标记"。如表 1.5 中表明的那样，很容易将这些组排序。

表 1.5

位	PG4	PG3	PG2	PG1
P_1				X
P_2			X	
D_1			X	X
P_3		X		
D_2		X		X
D_3		X	X	
D_4		X	X	X
P_4	X			
D_5	X			X
D_6	X		X	
D_7	X		X	X
D_8	X	X		
D_9	X	X		X
D_{10}	X	X	X	
D_{11}	X	X	X	X

1.25 假设题 1.24 中的单错校验码用在奇校验系统中，D_{11} 作为最高位排列的数据为：

D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1
1	0	1	0	0	1	1	1	1	0	0

被编码的序列是什么？

解

D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
1	0	1	0	0	1	1	1	1	1	0	1	0	1	1

P_4 的输入是 1，因为这个组的成员（D_5 到 D_{11}）有偶数个 1。对于其他的奇偶校验位的输入可重复这个过程。

1.26 如果接收到的序列是 100001111111011，题 1.25 的系统会采取什么措施？

解 注意：接收到的数据有两位（D_9 和 D_2）是错误的。然而，因为系统被设计成只要检测出单错，系统就会作出反应，似乎这就是其原因。我们可以看到，只要奇偶校验组 PG4 有误，检测电路就会"假定"仅属于 PG4（P_4）的位是惟一的错位。它会将 P_4 改为 0，听任所接收的信息仍然错误。

1.27 在 1.6 节中讨论的是 ARQ 编码技术。表明这种技术会检测到编码字中的所有奇数次错误和一些偶数次错误。用编码字 11001010 作为例子。

解 编码的关键是在每个字中准确地发送 4 个 1。单错一定会使其中的一个 1 变成 0 或者其中的一个 0 变成 1，并且在这两种情况下，1 的数目就会从 4 改变。对于任何奇数次错误，从 1 到 0 的转换次数总是与从 0 到 1 的转换次数不同，并且总会导致错误的 1。在偶数次错误的情况下，1 到 0 的转换次数与 0 到 1 的转换次数相平衡，因此保持 1 的数目为常数从而会导致错误检测失败。例如见下表，其中发射的编码是 11001010。

收到的编码	错误次数	1 的个数	
11001110	1	5	
10001010	1	3	
11110010	3	5	
11111110	3	7	
10100010	3	3	
10011010	2	4	(没有检测到)
11111010	2	6	

补 充 习 题

1.28 将二进制数 100110111 转换成十进制数，八进制数和十六进制数。

1.29 将十进制数 416 转换成二进制数，八进制数和十六进制数。

1.30 将十进制数 0 到 10 写成以 4 为基数的数。

1.31 将十进制数 250.5 表示为以 7 为基数的数和以 16 为基数的数。

1.32 将以下的数转换成它们相应的十进制数。

(a)1032.2_4　(b)0.342_6　(c)60_7　(d)188_{12}

1.33 十六进制 AB6 转换成二进制数，八进制数和十进制数。

1.34 将 205(十进制)表示成二进制数和十六进制数。

1.35 将 BCD 数 10010110 转换成十六进制数。

1.36 给定十六进制数 $ A38，将其转换成八进制数。

1.37 给定十六进制数 $ D3C，将其转换成 BCD 数。

1.38 给定八进制数 1216，将其转换成 BCD 数。

1.39 将八进制数 27 和 42 相加并以二进制数表示结果。

1.40 求出 - 119 的 2 的补码，假设 10 位二进制字长。

1.41 采用 2 的补码方法从 10100 中减去 100010。

1.42 将 - 105(基数为 10)和 - 56(基数为 8)相加。注意：两数都是负的。使用 9 位(包括符号位)，用二进制数的 2 的补码运算。将你的结果以十六进制表示。

1.43 从 736_8 中减去 583_{10}。以二进制数的 2 的补码进行运算并且将结果表示成基数为 6 的补码。

1.44 求 $ 0B7(十六进制)的 16 的补码。

1.45 将 1001101100011(格雷码)转换为二进制。

1.46 将 1100101101101(二进制)转换为格雷码。

1.47 一个轴编码器与图 1-2(b)中所示的相同，以格雷编码分成 32 个相等的扇区。如果轴顺时针旋转 36°，确定输出的格雷码。

1.48 将格雷码数 101101101011 表示成等值的十六进制数和二进制数。

1.49 在 ASCII 键盘上，如果按下了键 ADD 21，那么计算机输入什么二进制信号？

1.50 将数据 1011 按照表 1.4 的规则正确地编码成广单检错纠错码。受噪声污染后，接收到的数据位 D_3 和 D_2 是错误的。接收器要求只有 1 位错，并且处理相应的码组。给出完整的发送和接收字并表明接收器所做的处理。

补充习题答案

1.28 311(十进制)，467(八进制)，$ 137(十六进制)

1.29 110100000(二进制)，640(八进制)，$ 1A0(十六进制)

1.30 0，1，2，3，10，11，12，13，20，21，22

1.31　$505.333_{\ldots 7}$, \$ FA.8

1.32　(a)78.5　(b) 0.62037　(c) 42　(d) 248

1.33　101010110110(二进制), 5266(八进制), 2742(十进制)

1.34　11001101, \$ CD

1.35　\$ 60

1.36　5070_8

1.37　0011 0011 1000 1000

1.38　0110 0101 0100

1.39　$71_8 =$ 111001

1.40　1110001001

1.41　结果为负的值为 01110

1.42　结果为负。二进制的值是 10010111 与 \$ 97 相等

1.43　-105(以 10 为基数);大小为 5303_6

1.44　\$ F49

1.45　1110110111101

1.46　1010111011011

1.47　编码器转了 3.2 个扇区;如果第一个数为 0,格雷码将是 00010(需要 5 位)

1.48　\$ DB2, 110110110010(二进制)

1.49

A	D	D	空格	2	1
(1000001)	(1000010)	(1000010)	(0100000)	(0110010)	(0110001)

1.50　发送的字是 1011110,接收的字是 1101110。接收器会认为 D_1 是错误的,并且纠错后的数据是 1100,这是错误的

第 2 章　组合逻辑 I 的设计

2.1　组合逻辑

在任何时刻，所有输出状态仅仅取决于此时的输入值而与先前任何输入状态无关，这类数字电路被定义为组合逻辑电路。可将组合逻辑电路看成是具有 N 条输入线和 P 条输出线的黑箱，每一条线都具有传输数字(或逻辑)功能，它仅仅具有两个可能的值，通常标记为 1 和 0 或者是真和假。

从输入和输出关系的文字或符号说明开始，逻辑设计者的工作是用称作门的基本逻辑器件进行恰当的互连所组成的电路来填“黑箱”。

2.2　真值表

真值表是符号化地表示逻辑函数的一种便利的方法。将输入状态值的所有可能状态(通常以二进制升序方式)放在表格中，并且将与每个惟一的输入组合相对应的输出变量列在单独的列中，将其分配给每一个变量。因此真值表组成了要设计的组合逻辑的完整说明。

为了图示特定问题的真值表的使用，考虑将 4 位格雷码转换为二进制数的逻辑块的情况，要设计的黑箱如图 2-1 所示。



图 2-1

下标注明序列中每一位的位置，4 是最高位，1 是最低位。二进制码与格雷码的关系在 1.5 节中给出，用列表可以构建真值表。

真值表的构建

参考表 2.1。

1. 建立表，每一个输入和输出变量具有单独的列。

2. 以二进制升序的方式列出输入值的每一个可能的组合，填写左边的列(见例 2.1)。如果有 N 个输入变量，就会有 2^N 行，每一行都指明一个惟一的组合。

3. 对于每一行，填充一个相应的所要求的 P 个输出变量。

具有 N + P 列和 2^N 行的**真值**表，是组合逻辑的完整的符号说明，在本例中它表示了二进制码和格雷码之间的关系。

例 2.1　以下介绍在真值表输入侧使用的简单方法，即列出 N 个二进制变量值的所有可能组合。(参考表 2.1)

表 2.1

输入				输出			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

输入变量值的所有可能组合

1. 有 2^N 行(总是偶数)。
2. 最右边列(表中为 G_1)包含第 1 行以 0 开始进行 1 和 0 的交叉:0,1,0,1,0,1,…
3. 左边第 2 列由两个 0 接着两个 1 重复交换组成:0,0,1,1,0,0,1,1,…
4. 第 3 列从 4 个 0 开始,接着是 4 个 1,重复交换。
5. 总而言之,每 k 列有 $2^{(k-1)}$ 个交换的 0 和 1。

在很多情况下,从特定问题构建真值表是非常困难的,因此一定要经常详细研究逻辑所完成的过程才能发现某种形式。二进制加法器的实现说明了这个问题。在这种情况下,设计者能重复相加 2 个二进制数。既然二进制是位置计数法,需要研究单位数加法是显而易见的,因为总的过程对于每一位都是相同的。因此一位有效的过程对于其他位也是有效的,同时仅仅需要复制已经设计出的硬件来处理多位数。

产生的求和位决定于相加的 2 个数和前面低位数出现的进位。设计者会总结出,二进制的求和位的产生涉及到 3 个输入变量:X 位、Y 位和进位输入位 C_i。因此会有 2 个输出变量,除了求和位 S,一定要产生进位输出位 C_o 作为下一个高位的进位输入位。当确认了所有满足要求的变量时,逻辑的实现可以通过黑箱符号化来表示,如图 2-2 所示。

一旦理解了单位数的加法过程,可以通过真值表进行描述(表 2.2),它列出了所有可能的输入组合和相应的输出。现在我们有了组合逻辑的精简的符号说明,将其设计到黑箱中。

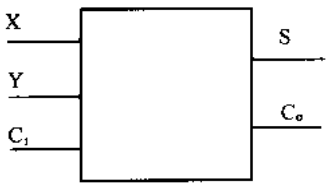


图 2-2

表 2.2

输入			输出	
X	Y	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.3　布尔表达式和基本逻辑函数

如上所示，组合逻辑的完整设计说明能以真值表的形式表达，也可将真值表中所有信息转化为一组逻辑表达式；每个逻辑表达式针对一个输出变量。重要的是可以利用称作布尔代数的大量数学运算构成的有效的逻辑函数结构，它能够处理仅具有两个值的变量(通常称为真或假)，我们可以认为真值表中1和0分别对应于逻辑代数中的真和假。

布尔代数表达式的结构

布尔表达式有以下的基本特点：

1．每个表达式具有这样的形式：F＝f(A，B，C，…)，其中F，A，B，C等是变量。

2．所有的变量都是逻辑变量，它们的特点是仅具有两个可能的值，真或假，高或低，1或0等。

3．表达式中逻辑变量通过逻辑操作符或运算符的方式联系。它们是逻辑等、逻辑与、逻辑或和逻辑非。

逻辑等

具有相同真值的两个逻辑变量是相等的。逻辑等的符号表示是等号(＝)。

例2.2　表达式A＝B意味着如果A是真，则B必须也是真，如果A是假，则B必须也是假，不会存在其他的可能性。注意：逻辑变量A、B可分别代表复杂逻辑表达式。

布尔与函数

与函数定义为：有且只有所有输入量为真时输出为真，逻辑与的符号是点(·)。

例2.3　表达式F＝A·B，表明只有A和B都为真时F为真。通常将点省掉，所以可以将其简写为F＝AB。与函数可用真值表表示为：

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

布尔或函数

或函数定义为当输入中的任何1个或多于1个为真时输出为真。逻辑或的符号表示为加号(＋)。

例2.4　表达式F＝A＋B表明A或B中的任1个或都为真时，F为真。

用真值表表示或函数：

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

逻辑非

表达式A′意味着如果A为真，则A′为假，反之亦然。

A	A′
真	假
假	真

A	A′
1	0
0	1

注意：在实际中，可用 $\bar{A}$ 或 A′表示逻辑非，使用字处理器的人喜欢用后者。

例 2.5　在表达式 F= A′B 中，如果 A 为假并且 B 为真时 F 为真。

异或函数

称作异或的运算符，虽然对于逻辑设计工程师非常有用，但严格说来，它并不是基本逻辑函数。它仅具有两个输入变量，当且仅当一个输入为真另一个输入为假时，输出为真。

经常将异或函数称作 XOR，它的符号是⊕。

例 2.6　异或函数 F = A⊕B 的真值表表示为

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

运算等级

正如在任何算法中一样，运算顺序是非常重要的。对于布尔运算，运算等级如下排列：

逻辑非　　最高
逻辑与　　↓
逻辑或　　↓
逻辑等　　最低

如常规代数的方式一样，使用括号可以改变运算执行顺序，首先运算最内层的括号，然后处理外面的括号，直到清掉所有的括号。

读布尔方程式

假设 F = AB + B′C′ + A′(C + D′)，如下读这个方程式：

(a) 如果方程式的右边(RHS)为真，F 为 1(真)。(使用等式关系)

(b) 如果 AB 或 B′C′或 A′(C + D′)中的任何组合为 1，RHS 为 1。(使用或的定义)

(c) 如果 A 和 B 都是 1，AB 为 1。(使用与的定义)

(d) 如果 B′和 C′都为 1，则 B′C′为 1。

如果 B 为 0 则 B′为 1。

如果 C 为 0 则 C′为 1。

因此，如果 B 和 C 都为 0，则 B′C′为 1。(使用逻辑与和逻辑非)

(e) 如果 A′，并且(C − D′)也为 1，则 A′(C + D′)为 1。

若 A 为 0，则 A′为 1。

若 C 为 1 或 D 为 0 或两种条件都满足则(C + D′)为 1。因此，如果 A 为 0 并且 D 为 0 或 C 为 1(或二者同时满足时)，则 A′(C + D′)为 1。

提示：要弄清括号是如何动态影响表达式的值的。例如，同学们应该明确 A′C + D′ 与 A′(C + D′)或 A′(C + D)′是完全不同的。

2.4 布尔表达式与真值表的关系

再次考虑所设计的用来完成1位二进制加法的模型。图2-2中,黑箱中的电路完成了二进制数输入X、Y和来自前面模型的进位输入位C_i的求和。有两个输出变量:和S与进位输出位C_o。为方便起见,在表2.3中再次写出表明了发生在黑箱中的加法过程的真值表。

表2.3

输入			输出	
X	Y	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

为了在布尔表达式中重新获得真值表中的信息,我们对两个输出列中的每一列写出一个表达式。注意:真值表中的每一行对应于输入变量的惟一组合。例如:最后一行对应的情况是X为1,Y为1并且C_i为1。如果将1等价于真,那么从真值表的末行可以看出S(和C_o)为真的一个条件是X、Y和C_i都为真,因此,从例2.3中可以得到

$$S = XYC_i \tag{2.1}$$

布尔与函数要求涉及到的所有变量都为真。为了在具有0的项的真值表中运用这个函数,我们使用例2.5中所示的逻辑非。因此可知写出真值表中的每一行的与表达式是可能的。如表达式

$$S = X'Y\,C_i' \tag{2.2}$$

对应于表2.3中的第三行,并可将其读做"当(如果)X′、Y和C_i'都为真时S为真"。等效表述为"当X和C_i'为假且Y为真时S为真"。

从真值表中我们看出,有多个输入情况(行)相应于输出为1(真)。因为我们看到相对于第2、3、5或8行的任何一个物理情况发生时和(S)为1(真),这里证明布尔或函数是有用的。用代数表示为

$$S = X'Y'C_i + X'Y\,C_i' + XY'\,C_i' + XY\,C_i \tag{2.3}$$

同样可将输出C_o表达为

$$C_o = X'YC_i + XY'\,C_i + XY\,C_i' + XY\,C_i \tag{2.4}$$

这两个表达式综合考虑,包含了真值表(表2.3)的所有信息,因为它们指明了在两个输出列中哪一个输入组合将产生1。输入列非1即0;因为布尔代数是二值的,它们一定没有其他值。

例2.7 从真值表输出列为0的所有行写出布尔表达式是可能的。对于正在考虑的加法器,所得的布尔表达式是对于S′=1和C_o'=1(等效于S和C_o为0)。这样便得到了

$$S' = X'Y'C_i' + X'YC_i + XY'C_i + XYC_i'$$

$$C'_0 = X'Y'C'_i + X'Y'C_i + X'YC'_i + XY'C_i$$

2.5 逻辑框图

画出要设计的并且用图来说明的逻辑框图常常是有帮助的。使用标准逻辑符号的原理图进行从逻辑方程式到功能电路的转换也是非常有用的。

基本函数及其符号

完成基本或和与的逻辑功能元件称作门电路，可用它们集成构建极其复杂的逻辑功能。它们的符号在图 2-3 中画出，有 3 个输入端，但实际中可根据需要使用任何数目的输入。

对于逻辑设计者，第 3 个基本符号虽然没有被普遍接受，但它们也是非常有用的，常用斜线标记 * 表示逻辑非(见图 2-4)。



F = A+B+C　　H = WXY

逻辑或　　逻辑与

图 2 3

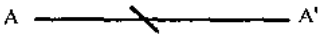


图 2-4

异或

如前边 2.3 节所提到的，异或不是基本函数；其逻辑用布尔形式表达为

$$F = AB' + BA' \tag{2.5}$$

并且正如我们能从例 2.6 中的真值表所看到的一样，可用两个与门和两个或门来合成它。异或函数经常出现，因此也分配给它 1 个标准逻辑符号(见图 2-5)。

例 2.8　可用异或门来产生逻辑非，如图 2-6 所示。

假设逻辑变量 A 通过逻辑门后可以变或者不变，它决定于作为控制开关的第 2 个变量的值。通过观察例 2.6 的真值表，可以很容易地观察到这种特性，即当 B = 0 时，F 与 A 相等；当 B = 1 时，F 为 A 的逻辑反。



F = A + B

图 2-5





图 2-6

组合符号

合并基本逻辑符号，图形化地获得逻辑功能图是可以的。例如可将方程式 F = A′BC + AB′D + A′B′CD′ 用图 2-7 描述。

另一方面，有时需要从给定的逻辑框图中获得基本的布尔方程式。考虑图 2-8 中的电路。输出变量 F 由或门 G_1 产生，它有 3 个输入端：变量 E 和与门 G_2、G_3 的输出。为方便起见，对于一个门和它的输出使用同样的设计。因此，对照上下文，可将 G_2 看作是由这个门产生的输出变量，记住这一规则，从框图中直接写出逻辑方程式是非常容易的事情。

* 图 2-4 所示的符号仅仅是逻辑符号；其硬件实现在 4.3 节中讨论。

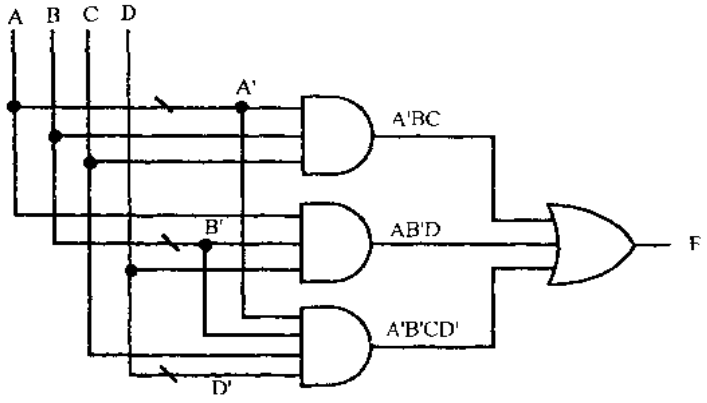


图 2-7

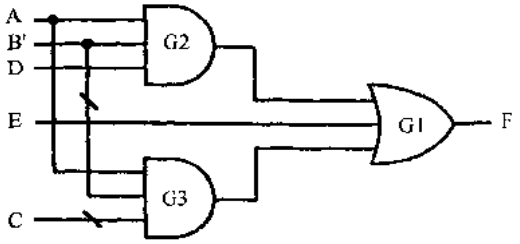


图 2-8

从左向右我们有

$$F = E + G_2 + G_3$$

$$G_2 = AB'D$$

$$G_3 = ABC'$$

简单的算术替代得到

$$F = E + AB'D + ABC'$$

事实上，逻辑框图的重要性在于数字电路制造者根据它们(如或门)所完成的逻辑来设计电子电路；电路符号通常非常接近于逻辑符号或逻辑符号的变化。因此，可用与逻辑符号相匹配的门电路来估计逻辑框图的硬件需求。另外，逻辑图中符号间的连接能够直接表明硬件元器件间的接线，既然物理器件的互连在不同接口处需要遵守兼容规则，逻辑图也成为设计者识别和解决潜在连接问题的一种重要手段。

习 题 解 答

2.1　一黑箱能接收 4 个数据位并能创建另外 3 个奇偶校验位，用作表 1.4 定义的单错校验汉明编码，创建该黑箱的真值表。将数据位称作 D_4, D_3, D_2, D_1，奇偶校验位表示为 P_3, P_2, P_1。

解 ☞ 用例 1.2 所示方法，用所有可能的输入组合填写真值表的左边的列，然后根据表 2.4 所定义的原则给 3 个输出变量分配值。例如，考虑第 4 行：0011。因为 $D_1 = 1, D_2 = 1, D_4 = 0, D_3$ 不是第 1 组的成员，可见第 1 个校验组有偶数个 1。为取得奇校验，必须添加 1，通过使校验位 P_1 为 1 来实现。继续分析，D_1 是第 2 个校验组的惟一成员，它只有一个 1，并且该组的校验为奇校验。因此，$P_2 = 0$。D_2 是第 3 个校验组中惟一的 1，因此，$P_3 = 0$。对每行继续这一过程直到完成表 2.4 所示的真值表。

表 2.4

D_4	D_3	D_2	D_1	P_3	P_2	P_1
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1
1	1	1	1	0	0	0

2.2 大多数计算器和电子表利用 7 段译码显示输出，这些显示的每一数位都由 7 段发光半导体材料组成(或吸光液晶)，如图 2-9(b)中所示排列。给这些段有选择性的供电就实现了所需数位的显示。例如，如果要显示十进制数 2，那么就给 a, b, g, e 和 d 段供电，取 BCD 数作为输入创建逻辑真值表，用 7 个输出驱动相应的显示位。

解 ☞ 以二进制递增的方式将相应的十进制数 0～9 排列在输入侧构建真值表(表 2.5)，输出侧列出了要被点亮的段来形成所选的数位；1 表明该段被激活(可视)。

提示：因为输入是 BCD 数，在输入侧有 6 个二进制组合未被使用。

表 2.5

十进制数位	输入				输出						
	W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

2.3 从布尔表达式中如何重构真值表。对应于下式构建真值表：

$$F = A'B'C' + A'BC' + AB'C' + AB'C$$

$$G = A'BC + ABC'$$

提示：这是获得表达式(2.3)和(2.4)的相反过程。

通用方法如下：

1. 以二进制升序方式列出独立变量的所有可能的组合，这样便构建了真值表的行。
2. 对每个独立变量创建输出列。
3. 对每个输出列从布尔表达式中确定哪些行中该置 1。
4. 用 0 填写余下的输出列。

解 ☞ 如果考察 F 的方程，我们看到有 3 个独立变量 A, B, C 和 4 个与函数组，其中每一组都对应于真值表中输出列 F 为 1 的一列。在每一个与函数组中，用 1 来替代直接变量(无撇号的)，0 替代逻辑取反变量，从而得到所需要的行的编码。因此，F = 1 的 4 行对应于输入变量组合 ABC = 000, 010,

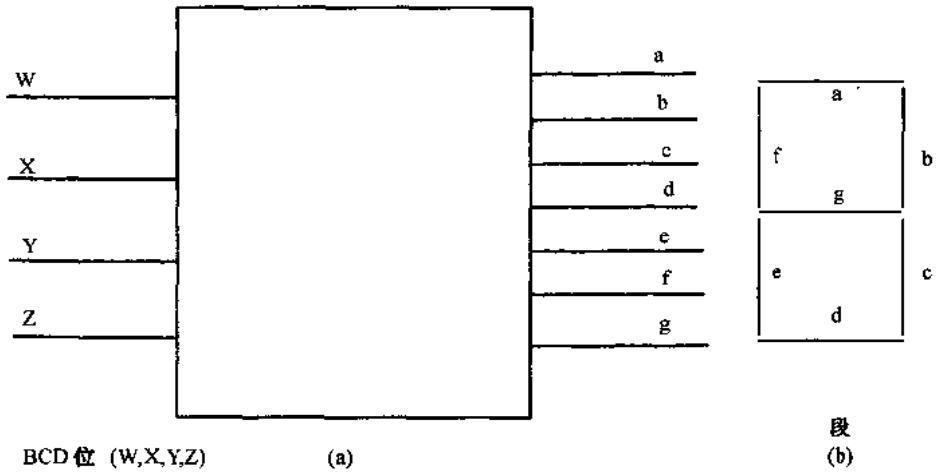


图 2-9

100 和 101。采用同样的方法来确定对应于 $G=1$ 的行后，可很容易的完成真值表(见表 2.6)。

表 2.6

输　入			输　出	
A	B	C	F	G
0	0	0	1	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

2.4　用基本布尔函数逻辑与、逻辑或和逻辑非写出异或函数 A⊕B 的布尔表达式。

解　从表 2.6 中给出的真值表中可直接写出所要的函数：

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

如果 A 和 B 不为真或者 B 和 A 不为真，那么函数 $F=A\oplus B$ 为真。异或的主要特点是它能从普通的或函数中辨认出如果 A 和 B 都为真，则 F 不是真。

以符号形式表示，$A\oplus B=AB'+BA'$。

2.5　另外一个通用的函数为异或非，$\overline{A\oplus B}$。可见其布尔表达式为 $AB+A'B'$，并且该函数定义了逻辑等。

解　构建真值表(见表 2.7)。

表 2.7

A	B	A⊕B	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

可见如果 A 和 B 有同样的值(逻辑相等)则函数$\overline{A \oplus B}$为真。从真值表中可得

$$\overline{A \ominus B} = AB + A'B'$$

提示:应用逻辑代数理论可获得同样的结果(参考问题 3.4)。

2.6 对表 2.8 所示真值表构建相应的逻辑表达式。

解 ☞ 第4、6 和 8 行所指的组合 F 为真,第 1、2、6、7 和 8 所指的组合 G 为真。

$$F = A'BC + AB'C + ABC$$
$$G = A'B'C' + A'B'C + AB'C + ABC' + ABC$$

表 2.8

A	B	C	F	G
0	0	0	0	1
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

表 2.9

A	B	C	W	X	Y
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	1	1

2.7 由以下输出方程式确定系统的真值表。

$$W = AB'C + A'B'C + ABC'$$
$$X = A'B'C + AB'C + ABC' + ABC$$
$$Y = A'B'C' + A'B'C + AB'C + AB'C' + ABC$$

解 ☞ ABC 为以下的情况时 W 为真:001、101 和 110。ABC－001、101、110 和 111 时 X 为真。ABC=000、001、100、101 和 111 时 Y 为真,见表 2.9。

2.8 要求设计的逻辑系统有两输出端(F, G)和三输入端(W, X, Y)。如果奇数个输入为真,则输出之一 F 为真。仅有 1 个输入为真,则另 1 输出 G 为真。画出真值表并写出相应的布尔表达式。

解 ☞ 从已知的输入个数(3)创建真值表的输入端,运用设计标准获得每一行的输出值,如表 2.10 所示。依次写下对应于输出为 1 的输入行组合(逻辑与项)来确定输出表达式:

$$F = W'X'Y + W'XY' + WX'Y' + WXY$$
$$G = W'X'Y + W'XY' + WX'Y'$$

2.9 根据表 2.11 所示的真值表,用输出列包含 0 的组创建指导型布尔表达式。

解 ☞ 在输出列中对应于 0 的表达式为

$$F' = A'B'C' + A'BC' + A'BC + AB'C$$
$$G' = A'B'C' + A'B'C + AB'C' + AB'C + ABC'$$

表 2.10

W	X	Y	F	G
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	0

表 2.11

A	B	C	F	G
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

2.10 以下函数表示了典型的异或结构：

$$F(A, B, C, D) = (A \oplus B) \oplus (C \oplus D)$$

构建其真值表。

解 系统地对每一组填写输出列就可获得合乎要求的真值表(表 2.12)。

仔细观察真值表可看出：仅当输入中 1 的个数为奇数时，F 为 1。这是异或函数的共有特点，这对于奇偶发生和错误检测系统的实现是非常有用的。

表 2.12

A	B	C	D	A⊕B	C⊕D	F
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	1	1
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	0
0	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	0	0	0

2.11 写出题 2.2 中 7 段译码器的布尔表达式。

解 参考题 2.2，想到真值表中的行对应于与函数的组并且每个输出列对应于这些与函数的或。

a = W′X′Y′Z′ + W′X′YZ′ + W′X′YZ + W′XY′Z + W′XYZ′ + W′XYZ + WX′Y′Z′ + WX′Y′Z

b = W′X′Y′Z′ + W′X′Y′Z + W′X′YZ′ + W′X′YZ + W′XY′Z′ + W′XYZ + WX′Y′Z′ + WX′Y′Z

c = W′X′Y′Z′ + W′X′Y′Z + W′X′YZ + W′XY′Z′ + W′XY′Z + W′XYZ′ + W′XYZ + WX′Y′Z′ + WX′Y′Z

d = W′X′Y′Z′ + W′X′YZ′ + W′X′YZ - W′XY′Z + W′XYZ′ + WX′Y′Z′ + WX′Y′Z

e = W′X′Y′Z′ + W′X′YZ′ + W′XYZ′ + WX′Y′Z′

f = W′X′Y′Z′ + W′XY′Z′ + W′XY′Z + W′XYZ′ + WX′Y′Z′ + WX′Y′Z

g = W′X′YZ′ + W′X′YZ + W′XY′Z′ + W′XY′Z + W′XYZ′ + WX′Y′Z′ + WX′Y′Z

2.12 画出题 2.1 的奇偶发生器的输出 P_1 的逻辑图。

解 相关的输入可直接从真值表的行中获得。参考图 2-10。

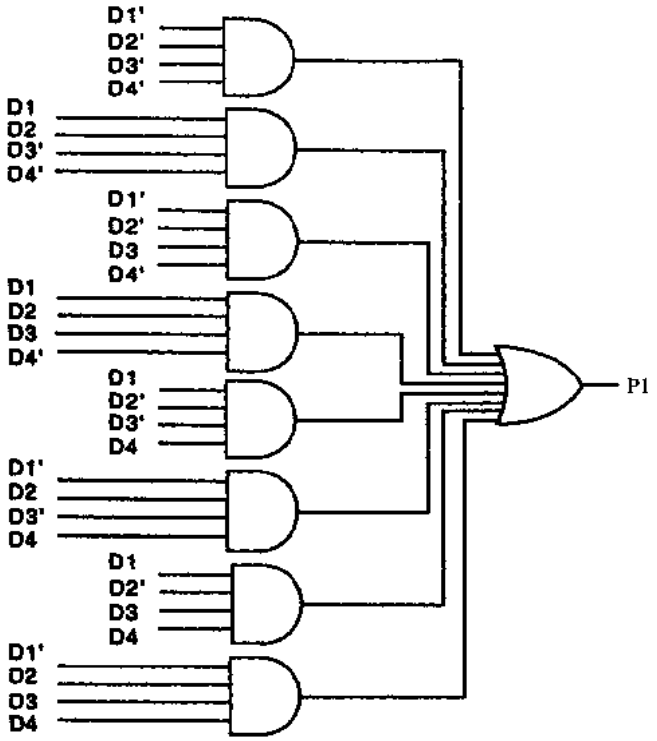


图 2-10

2.13　画出图 2-2 中二进制位加法器的逻辑图。

解　两个相关的布尔表达式已从真值表[表达式(2.3)和(2.4)]中推导出。

$$S = X'Y'C_i + X'YC_i' + XY'C_i' + XYC_i$$

$$C_o = X'YC_i + XY'C_i + XYC_i' + XYC_i$$

观察图 2-11 所示图可画出逻辑图。

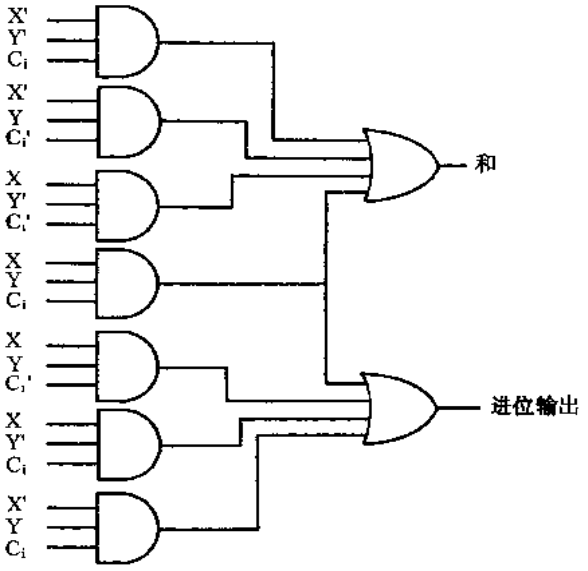


图 2-11

2.14　写出相应于图 2-12 所示逻辑图的布尔表达式。

解

运算输出	函数
G_1	AB'
G_2	$G_1 + C = AB' + C$
G_5	CD'
G_4	$G_5 + A' + B' = CD' + A' + B'$
$G_3 = F$	$D \cdot G_2 \cdot G_4 = D(AB' + C)(CD' + A' + B')$

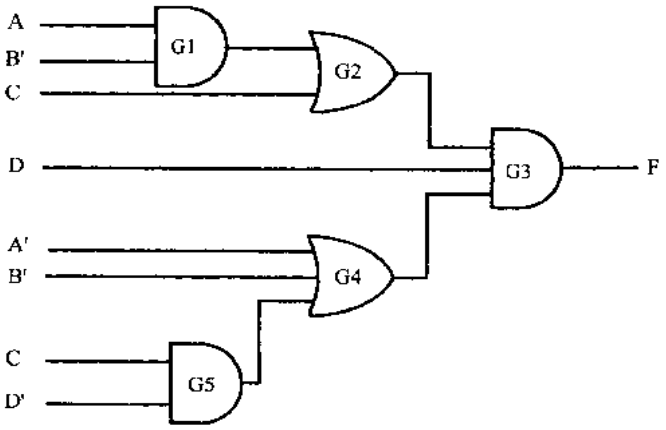
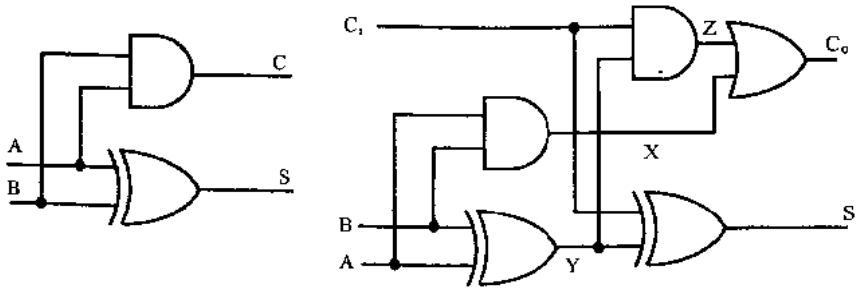


图 2-12

2.15　图 2-13(a)中所示逻辑图称作半加器，确定其真值表，同时给出由两个半加器组合形成的全加器(图 2-13(b))。确定图 2-13(b)逻辑的真值表，同时表明全加器确实能从低位数中接收进位，并且能产生下一个高位数所需的进位输出位。



(a) 半加器　　(b) 全加器

图 2-13

解　(a)半加器的真值表为

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

当 A 和 B 都为 1 时，C(进位位) = 1。仅当 A 或 B 为 1，并且不全为 1 时 S(和)为 1。可看到逻辑遵循单个位相加规则，只有在输入端 A 和 B 同时出现两个 1 时，产生进位。

(b)用以下中间变量画出全加器真值表(表 2.13),完成过程:

$$X = AB \qquad Z = C_i Y \qquad Y = A \oplus B \qquad S = C_i \oplus Y \qquad C_o = Z + X$$

表 2.13 表明,当两个或更多输入为 1 时产生进位输出位,只有当输入奇数个 1 时,和输出为 1。这满足 1.4 节中所描述的二进制加法规律。

表 2.13

A	B	C_i	X	Y	Z	C_o	S
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	1
0	1	1	0	1	1	1	0
1	0	0	0	1	0	0	1
1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0
1	1	1	1	0	0	1	1

2.16 在图 2-14 所示方框图中,方框代表逻辑器件,当且仅当 3 个输入中大多数为真时,输出为真。画出箱子相应的逻辑框图。

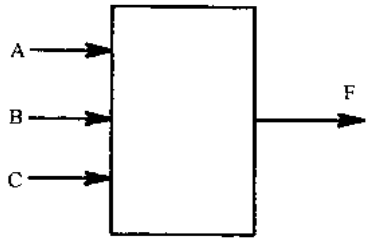


图 2-14

解 写出真值表。

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

通过观察得到 F 的布尔方程如下:

$$F = A'BC + AB'C + ABC' + ABC$$

从中可得到所要求的逻辑图,如图 2-15 所示。

2.17 在图 2-16 所示的电路图中,方框代表等同于题 2.16 所描述的逻辑器件。写出 G 的布尔表达式。

解 电路 1 的表现在第 2.16 题已描述:

$$F_1 = A'BC + AB'C + ABC' + ABC$$

电路 2 表现为 C′和 D 的或门,因为在一个输入端为直接连接 1:

$$\begin{aligned} F_2 &= CD \cdot 1 + C'D' \cdot 1 + C'D \cdot 0 + C'D \cdot 1 \\ &= CD + C'D' + C'D \\ &= C' + D \end{aligned}$$

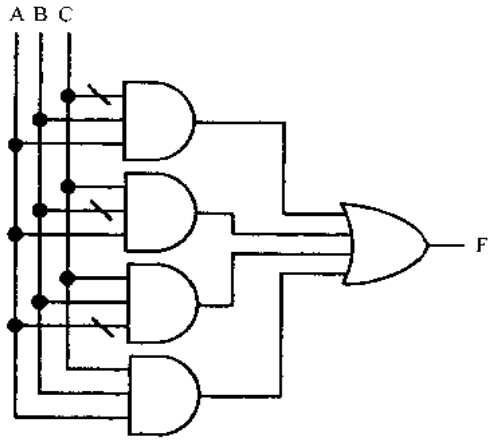


图 2-15

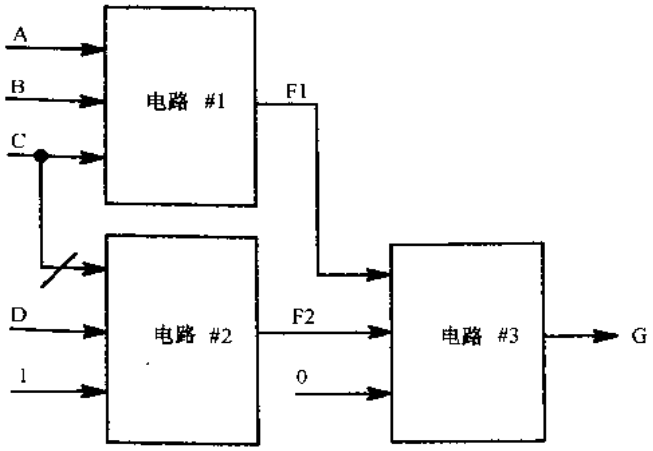


图 2-16

其实现可用真值表或参考第3章讨论的基本逻辑关系来确认。

电路3的功能表现为 F_1 和 F_2 的与门，因为一输入直接连接0。

$$G = F'_1F_2 \cdot 0 + F_1F'_2 \cdot 0 + F_1F_2 \cdot 1 + F_1F_2 \cdot 0$$
$$= F_1F_2 = (C' + D)(A'BC + AB'C + ABC' + ABC)$$

补 充 习 题

2.18　画出代表函数 $F = (A'B + AD')C$ 的逻辑框图，输入为A、B、C和D。

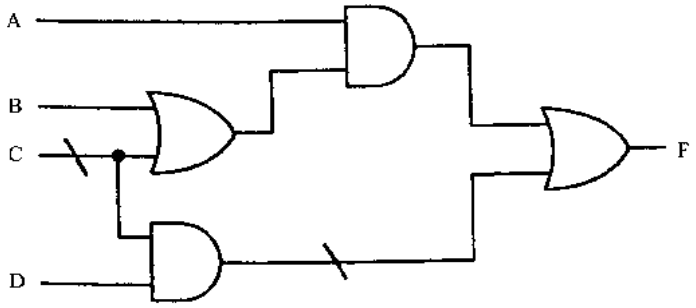


图 2-17

2.19 写出图 2－17 的逻辑图代表的布尔表达式。

2.20 创建实现两个 2 位数(A_1, A_0 和 B_1, B_0)乘法的电路的真值表。

2.21 写出第 2.20 题乘法器的布尔方程式。

2.22 画出第 2.21 题方程式的逻辑图。

2.23 为以下由布尔方程式指定的系统创建真值表，画出逻辑图：

$$F = A'B'C + A'BC + AB'C$$

$$G = A'BC' + A'BC + ABC'$$

2.24 为表 1.4 所描述的产生汉明码奇偶校验位的电路创建真值表和逻辑方程式。

2.25 为通过两个开关(A 和 B)能控制一个灯(L)的电路创建真值表和逻辑方程式。当 A 和 B 都为真时灯会亮(真)，并且两个开关中的任何一个能够独立开关灯。

2.26 确定图 2-18 所示逻辑图的布尔方程式和真值表。

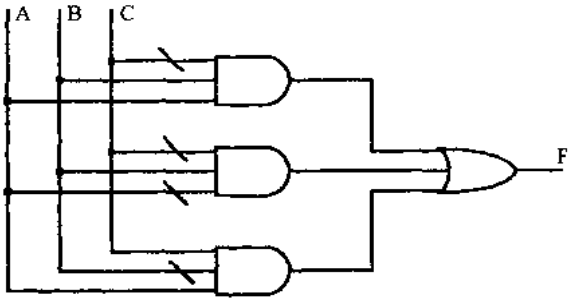


图 2-18

2.27 创建能完成操作 X－Y=D 的单个数位二进制减法器的真值表，按照与 2.4 节描述的用来研究单位二进制加法的相似的方式进行，用"借位"的概念代替"进位"，并且假设能借到位。

2.28 使用第 2.27 题创建的 1 位二进制减法器的真值表来得到输出的布尔方程式。

2.29 使用第 2.27 题创建的 1 位二进制减法器的真值表或第 2.28 题得到的方程式，画出相应的逻辑图。

2.30 一系统有三输入(A, B, C)和两输出(D, E)，其中当且仅当输入为真的数目为奇数时 D 为真，只要至少有两个输入为真时 E 为真。创建该系统的真值表。

2.31 创建图 2-19 逻辑图所描述的系统的真值表。

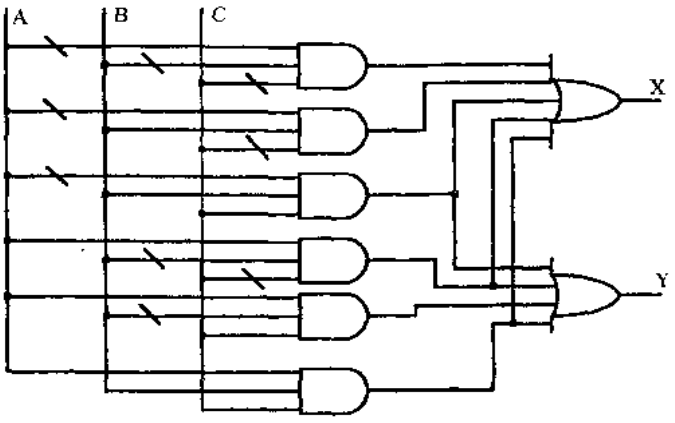


图 2-19

2.32 写出第 2.30 题逻辑的布尔方程式。

2.33 一数字系统有 4 个输出和 5 个输入，后者包括一个 4 位二进制数和一个加/减(U/D)位。输入数的范围介于与 3 和 12 之间相应的二进制数。若加/减(U/D)为 0，输出为下一个高阶二进制数，若加/减(U/D)为 1，输出为下一个低阶二进制数，序列自动停止。因此输入 12 在输出端会产生 3，输入 3 会产生 12。称输入为 A, B, C, D, 加/减，输出为 A_n, B_n, C_n, D_n，对所有可能输入组合创建真值表。

补充习题答案

2.18 见图 2-20。

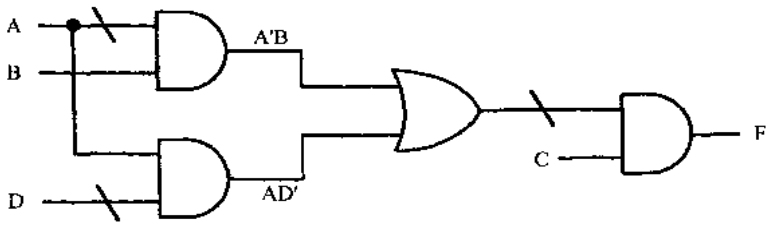


图 2-20

2.19 $A(B+C')+(C'D)'$

2.20 最大的两位数为 11(十进制为 3);因此最大的结果为十进制数 9(二进制 1001),对于这个结果需要 4 位,将其标记为 C_3, C_2, C_1 和 C_0。得到表 2.14 所示真值表。

表 2.14

输入				输出			
A_1	A_0	B_1	B_0	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

2.21

$C_3 = A_1A_0B_1B_0$

$C_2 = A_1A_0B_1B'_0 + A_1A'_0B_1B_0 + A_1A'_0B_1B'_0$

$C_1 = A'_1A_0B_1B'_0 + A'_1A_0B_1B_0 + A_1A'_0B'_1B_0 + A_1A'_0B_1B_0 + A_1A_0B'_1B_0 + A_1A_0B_1B'_0$

$C_2 = A'_1A_0B'_1B_0 + A'_1A_0B_1B_0 + A_1A_0B'_1B_0 + A_1A_0B_1B_0$

2.22 见图 2-21。注意:一些与门在多个表达式中出现,但仅需要实现 1 次。

2.23 见图 2-22。

2.24 既然奇偶校验位取决于输入位,所以用后者作为输入。真值表在表 2.15 中给出。

$P_3 = D'_4D'_3D'_2D'_1 + D'_4D'_3D'_2D_1 + D'_4D_3D_2D'_1 + D'_4D_3D_2D_1$

$+ D_4D'_3D_3D'_1 + D_4D'_3D_2D_1 + D_4D_3D'_2D'_1 + D_4D_3D'_2D_1$

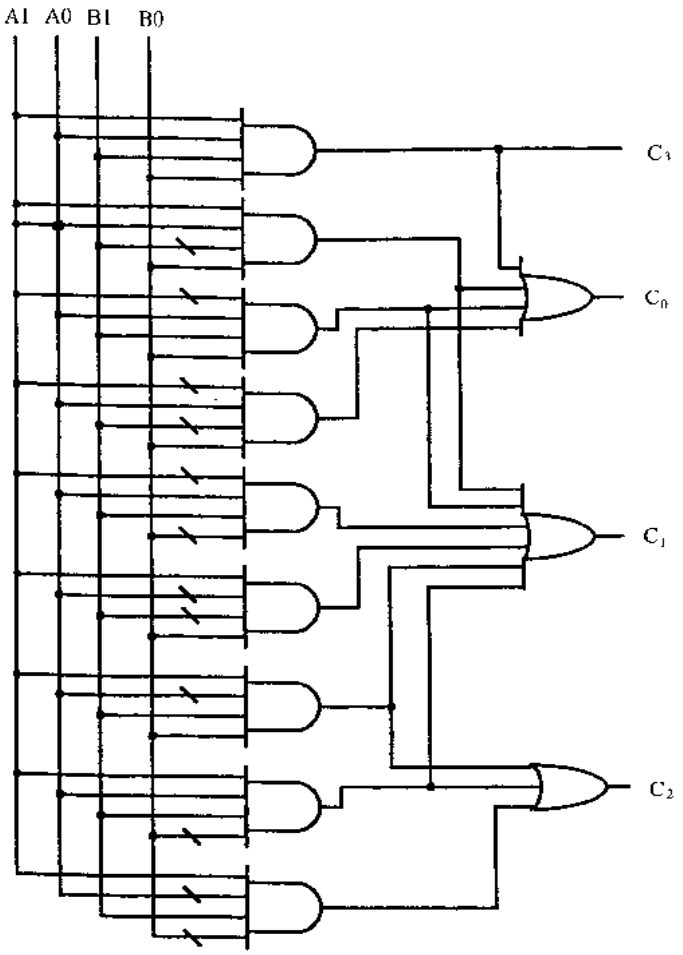


图 2-21

A	B	C	F	G
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

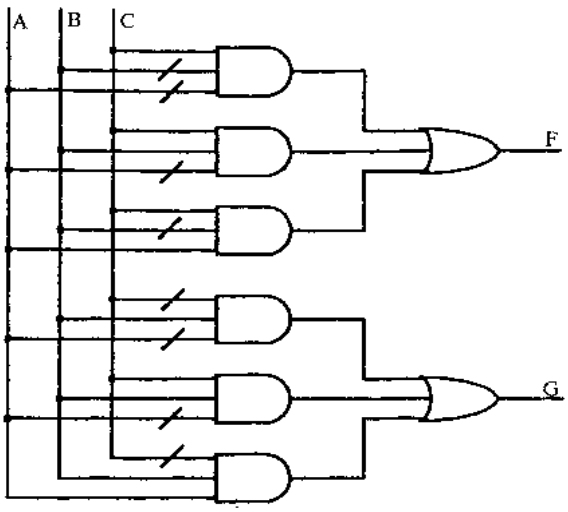


图 2-22

$$P_2 = D'_4D'_3D'_2D'_1 + D'_4D'_3D_2D'_1 + D'_4D_3D'_2D_1 + D'_4D_3D_2D_1$$
$$+ D_4D'_3D'_2D_1 + D_4D'_3D_2D_1 + D_4D_3D'_2D'_1 + D_4D_3D_2D'_1$$
$$P_1 = D'_4D'_3D'_2D'_1 + D'_4D'_3D_2D_1 + D'_4D_3D'_2D'_1 + D'_4D_3D_2D_1$$
$$+ D_4D'_3D'_2D_1 + D_4D'_3D_2D'_1 + D_4D_3D'_2D_1 + D_4D_3D_2D'_1$$

表 2.15

D_4	D_3	D_2	D_1	P_3	P_2	P_1
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1
1	1	1	1	0	0	0

2.25 L = AB;AB′或 A′B 必须将灯关掉(L′ = AB′ + A′B)。并且,1 个单开关操作产生 AB 或 A′B′来将灯打开。可用两个布尔方程中的任一个描述操作并产生表 2.16 给出的相应的真值表。

$$L = AB + A'B' \qquad L' = A'B + AB'$$

表 2.16

A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

2.26 F = ABC′ + A′BC′ + AB′C

真值表在表 2.17 中给出。

表 2.17

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

2.27 设 B_0 为“借位输出位”,也就是需要来自于高阶的借位。设 B_1 表示“借位输入位”,意思是已经借到的

位。表 2.18 中给出真值表。

表 2.18

X	Y	B_i	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

2.28

$D = X'Y'B_i + X'YB'_i + XY'B'_i + XYB_i$

$B_o = X'Y'B_i + X'YB'_i + X'YB_i + XYB_i$

2.29

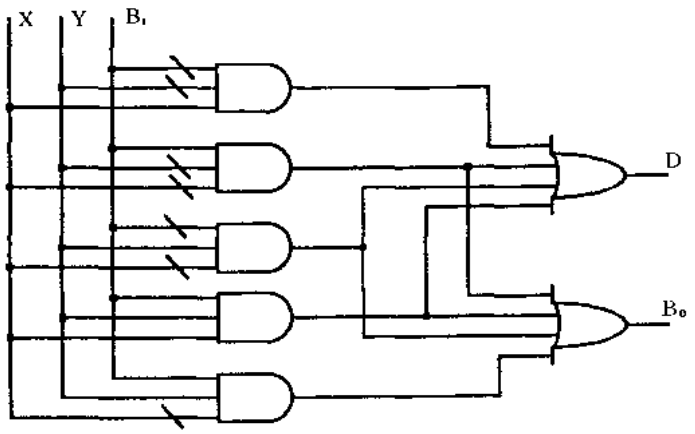


图 2-23

2.30

表 2.19

A	B	C	D	E
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.31

表 2.20

A	B	C	X	Y
0	0	0	1	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

2.32

$D = A'B'C + A'BC' + AB'C' + ABC$

$E = A'BC + AB'C + ABC' + ABC$

2.33

表 2.21

U/D	A	B	C	D	An	Bn	Cn	Dn
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	1	0	1
0	0	1	0	1	0	1	1	0
0	0	1	1	0	0	1	1	1
0	0	1	1	1	1	0	0	0
0	1	0	0	0	1	0	0	1
0	1	0	0	1	1	0	1	0
0	1	0	1	0	1	0	1	1
0	1	0	1	1	1	1	0	0
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	1	0	0
1	0	1	0	0	0	0	1	1
1	0	1	0	1	0	1	0	0
1	0	1	1	0	0	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	0	0	0	1	1	1
1	1	0	0	1	1	0	0	0
1	1	0	1	0	1	0	0	1
1	1	0	1	1	1	0	1	0
1	1	1	0	0	1	0	1	1

第3章　组合逻辑Ⅱ的设计方法

3.1　引言

如同第2章中所论述的，组合逻辑完整的设计说明可以用真值表的形式表达，并且真值表中的所有信息可用一组逻辑方程式表达。既然真值表和布尔变量都具有惟一的二值特性，所以我们可以利用已建立起的有用的布尔代数定理和简化定律，这些技术使我们能够处理一组逻辑方程式从而获得相应的表达式，这样经常实现极重要的简化电路。

真值表或与其相关的布尔方程式也可以用卡诺图的形式图形化地表示。这种方法允许设计者利用人类大脑的极为出色的图形认识能力来实现复杂布尔方程式的快速化简。

3.2　布尔代数基础

基本定律

第2章中给出的基本布尔运算的定义产生了几个有用的定律。在下面列举中，A是任何布尔(逻辑)变量，可取值1(真)或0(假)：

1. 任何逻辑变量和1进行或操作结果为1。

$$A+1=1 \tag{3.1}$$

2. 任何逻辑变量和0进行或操作结果保持不变。

$$A+0=A \tag{3.2}$$

3. 任何逻辑变量和1进行与操作结果保持不变。

$$A\cdot 1=A \tag{3.3}$$

4. 任何逻辑变量和0进行与操作结果为0。

$$A\cdot 0=0 \tag{3.4}$$

5. 任何逻辑变量和自身的与操作或者是或操作结果保持不变。

$$A+A=A \tag{3.5a}$$

$$A\cdot A=A \tag{3.5b}$$

6. 一个逻辑变量和它的非进行或操作，结果为1。

$$A+A'=1 \tag{3.6}$$

7. 一个逻辑变量和它的非进行与操作，结果为0。

$$A\cdot A'=0 \tag{3.7}$$

例3.1　借助于一个简单的电路模型可以很容易的理解和记住各种不同的布尔定律。如果将逻辑1认为是使路径接通，如的开关触点闭合，那么逻辑0就是一个非接触的开关触点。这看起来是合乎情理的。那么，每个与函数可以看作是正常情况下两个开关触点的串联连接，当对开关操作时，它们会闭合(变成导通)(见图3-1)，必须对开关A和B同时进行操作，灯(L)

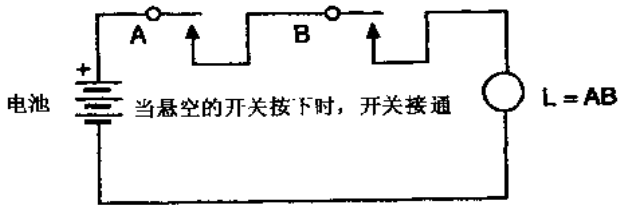


图3-1

才会亮(真)。如果B等于0(没操作,因此不导通),不管A是否操作,那么灯必定是灭的,这可以理解A·0=0 (定律4)。

同样可将或函数修改成正常情况下两个并联的开关(见图3-2)。单独关闭A或B,可使灯点亮。如果B等于1(激活或导通),那么无论A如何,灯一定是亮(真)的。这可以理解A+1=1(定律1)。

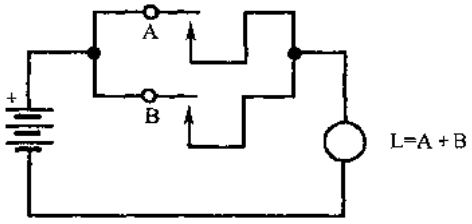


图3-2

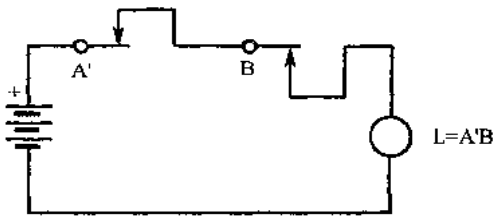


图3-3

逻辑非函数可以表示为一个正常情况下开关的闭合,当对其进行操作时,它不导通(假)(见图3-3)。在图3.3的电路中,必须清楚为了将灯点亮(真),B必须动作而A一定不动作。

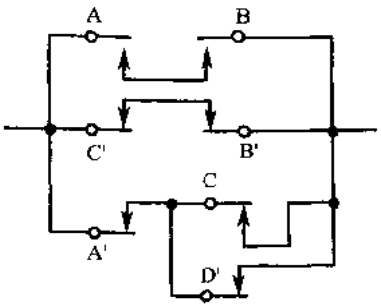


图3-4

正如我们所期望的,可以用正常情况下开关通和断的正确互连来表示任何组合逻辑函数。例如布尔表达式F=AB+B′C′+A′(C+D′)可以用图3-4所示的电路来模拟,当它导通时表示F为真。开关与逻辑控制柜类似,除了可以作为可视化辅助使用外,它还具有重要的历史意义。在固态器件和集成电路(IC)发明前,实际上逻辑控制由机电开关(继电器)构建成的,并且最早的计算机包含成千上万个由复杂线路精确连接成的设备。用作硬件或软件故障的术语"bug",追其根源回到1940年,当时计算机的故障是由一只死虫卡在一对继电器触点之间引起的!

分配率

$$A(B+C)=AB+AC \qquad \text{(和普通代数一样)} \tag{3.8}$$

交换率

$$AB=BA \tag{3.9(a)}$$

$$A(B+C)=(B+C)A \tag{3.9(b)}$$

结合率

$$ABC+ABC'=AB(C+C')=AB \tag{3.10}$$

这个定律对减化布尔表达式的项数是非常有用的。其依据是利用基本定律3和6,一个相关有用的定律是

$$A+AB=A \tag{3.11}$$

用分配率和基本定律1和3可很容易证明该式。

摩根定律

布尔代数中最重要的定律之一可以用两个相关联的形式表达:

$$AB=(A'+B')' \tag{3.12(a)}$$

$$A+B=(A'B')' \tag{3.12(b)}$$

学生们可能很熟悉另外一种等效的表达式：

$$AB=\overline{(\overline{A}+\overline{B})} \tag{3.13(a)}$$

$$A+B=\overline{\overline{A}\,\overline{B}} \tag{3.13(b)}$$

对于这些理论的证明，可以参考习题解答一节。摩根定律的重要性经常被忽视。如果设计者添加几个逻辑非，那么与操作可以改成或操作，反之亦然。这告诉我们在每个与函数里也有一个或函数，假设可使用足够多的逻辑非，那么由与逻辑实现的任何硬件同样的也可由或逻辑来实现。

通过穷举证明定律

利用穷举的方法可以很容易证明以逻辑方程式表达的布尔代数定律。

1. 构造一个 2^N 行的真值表，其中 N 是变量个数。
2. 给方程式两边的不同项分配单独的列。
3. 给每一个输入变量组合(行)进行这些项的值的计算。
4. 如果每一个输入组合若使方程式两边产生同样的真值，那么理论就被证明了。

例 3.2 用穷举法对分配率 A(B + C) = AB + AC 的证明在表 3.1 中给出。注意：对于输入变量的所有可能组合(也就是真值表中的每一行)，表 3.1 中框起来的两列是相等的。因此对应的布尔函数 A(B + C)和 AB + AC 一定在逻辑上相等。

表 3.1

A	B	C	B + C	A(B + C)	AB	AC	AB + AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

布尔代数运算中一个有用的定律是

$$A+A'B=A+B \tag{3.14(a)}$$

用穷举方法可以很容易地证明。

简单的变量替换产生了方程式(3.14(a))的补充形式，它也是非常有用的：

$$A'+AB=A'+B \tag{3.14(b)}$$

通过运算实现布尔表达式的化简

复杂表达式中通常含有多余的项，通过系统地应用基本定律和等式可以将多余项去掉。

例 3.3 化简(ab + bc′ + cd + bd′ + bc).

$f = ab + bc' + cd + bd' + bc$

$= ab + b(c' + c) + cd + bd'$ (分配率)

$= ab + b + cd + bd'$ (基本定律 6 和 3)

$= b + cd + bd'$ 方程(3.11)

$= b + cd$ 再一次用方程(3.11)

3.3 硬件实现

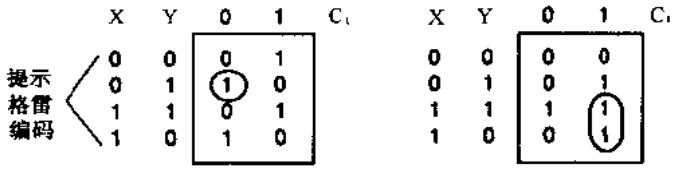
在硬件上实现组合逻辑设计的与和或操作是通过称作门的电子线路元件完成，它们的输入和输出是与逻辑状态真和假相联系的电平。通过正确地连接也可以用这些门来实现逻辑反。第4章的4.2节和4.3节详细讨论了硬件和逻辑间的关系。在这一点上，足以指出用布尔代数所获得的逻辑简化能够影响要求完成给定逻辑设计的硬件的数量。逻辑门用掉硅片上的空间，将布尔方程式处理成逻辑上等效的简化形式，能够极大地压缩集成电路的大小，并且因此使其速度有极大的提高，成本上有很大的降低。高效的代数简化需要技巧和经验，对于复杂的逻辑设计，最优的简化如同科学一样充满艺术性并且经常需要使用功能强大的计算机辅助设计软件。

3.4 卡诺图基础

卡诺图是利用人类大脑超乎寻常的图形识别能力，在布尔表达式的简化上作为一个强有力的工具。卡诺图与真值表直接相关，可从真值表中推导出卡诺图，如下所示：

行	输入 X	Y	C_i	输出 S	C_o
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

(a) 真值表



(b) S的卡诺图　　(c) C_o的卡诺图

图 3-5

1. 对每一个要求的输出逻辑变量创建1个卡诺图。

2. 沿着两个正交侧映射坐标轴，包含了一组完整的以格雷码形式排列的输入变量的所有可能组合。

3. 卡诺图中的每一项对应于真值表中每一行的输出值。

考虑与表2.2中定义的1位加法器的和以及进位输出位相关联的卡诺图。

在图3-5的卡诺图中，每一项对应于真值表中的1行并且每一个1对应于与卡诺图相关的输出变量的布尔方程中的1个与项。例如，在输出S卡诺图中圈起来的项具有的坐标是X=0，Y=1和C_i=0，这对应于真值表的第2行和S的布尔表达式的与项的$X'Y\,C_i'$(参考2.4节)。

格雷编码的目的是保证图中相邻的两项对应的输入在值上仅有1个变量不同。接下来会讨论这样安排的好处。

例3.4 C_o的布尔方程式可直接从相应的卡诺图中读出，仅仅通过写下卡诺图中每一个1的坐标并用逻辑或将其连接。因此通过观察图3-5中C_o的卡诺图就可以得到：

$$C_o = X'YC_i + XYC_i' + XYC_i + XY'C_i$$

用卡诺图来简化布尔表达式

在C_o卡诺图中圈起来的一对相邻的1对应于两组布尔表达式$XYC_i + XY'\,C_i$，结合律[方程(3.10)]表明可将其化简成一项(XC_i)，因为最初的两个组仅仅有1个变量(Y)不同。

通过对卡诺图的观察，将相邻的两个1认为是一个块，然后确定能够定义这个块的最少的

坐标变量，直接运用结合律。显而易见在图 3-5(c)中，第 2 列中有两个 1($C_i=1$)，对于这两个 1，X 也是 1。因此，相应的布尔项是 XC_i，删除了 Y，它是惟一独立的变量。它在圈起两个 1 的块中可以取两个可能的值。

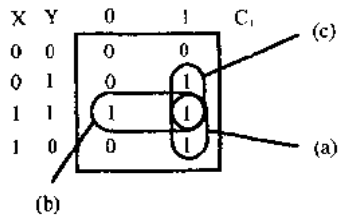


图 3-6

在 C_o 的卡诺图中有另外两个包含两个 1 的块，如图 3-6 所示。b 块产生了简化项 XY，删去了 C_i，c 块产生了 Y C_i，删去了 X。提示：块允许是重叠的；在本例中有一个 1 用了 3 次。

现在我们可以写出 Co 的简化布尔表达式：

$$C_o = \underline{XC_i} + \underline{XY} + \underline{YC_i}$$

块(a)　　块(b)　　块(c)

将它与例 3.4 中没有简化的表达式相比较，它们是完全相等的。

例 3.5　用卡诺图简化 C_o 的代数证明。基本定律 5(3.2 节)允许我们在例 3.4 中布尔方程式的右边将 XYC_i 项加两次而不改变它的正确性：

$$C_o = XYC'_i + XYC_i + XY'C_i + XYC_i + X'YC_i + XYC_i$$

使用分配律，

$$C_o = XY(C_i + C'_i) + XC_i(Y + Y') + YC_i(X + X')$$

因为 $A + A' = 1$(定律 6)，

$$C_o = XY \cdot 1 + XC_i \cdot 1 + YC_i \cdot 1$$

最后因 $A \cdot 1 = A$(定律 3)我们得到

$$C_o = XY + XC_i + YC_i$$

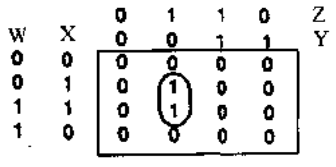
提示：卡诺图简化结果与图中含 1 项的圈法有关。

4 变量卡诺图

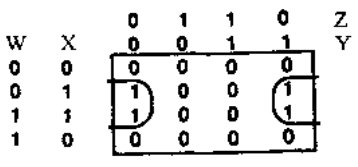
考虑下面的函数：

$$F(W, X, Y, Z) = XY'Z + X'Y'Z' + XZ + W'XZ'$$

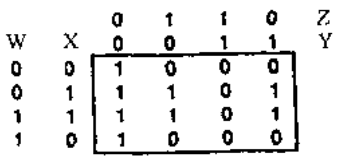
这个函数要求 4 个变量的卡诺图，它的坐标按格雷编码对安排，如图 3-7 所示。为了理解映射过程如何进行，我们来考虑单个的项。XY′Z 项表明 X = 1，Y = 0，Z = 1 而 W 是不相关的。值 X = 1 与中间两行相关，同时 Y = 0，Z = 1 惟一地与第 2 列相关。因此 XY′Z 项位于中间行与第 2 列的交接处，如同 3-7(a)中圈起来的 1 所定义的。同样 XZ′项与中间两行(X = 1)和第 1 及第 4 列(Z = 0)相关。这映射成了一个有 4 个 1 的矩形块，如图 3-7(b)所示。提示：可以将卡



(a) XY'Z 项



(b) XZ' 项



(c) F(W,X,Y,Z)

图 3-7

诺图看作是卷起来的管子，它的第 1 列和最后 1 列或者是第 1 行和最后 1 行相邻。图 3.7(c)表明了整个函数在 1 个卡诺图中的显示。

化简原则

研究图并找出足够多的相邻含 1 的方块，同时要遵守以下原则：

1. 任何圈或块必须并且仅仅包含 2^N 个 1，其中 N 是整数。

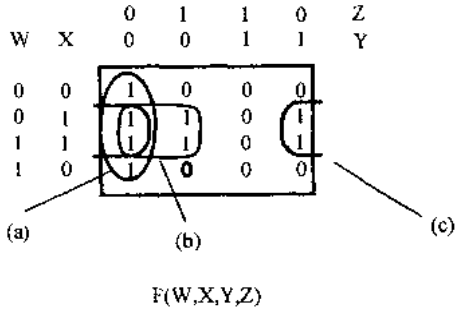
2. 在任何块中都不能有零。

3. 块可以重叠；也就是 1 可以被包含在多个块中。

4. 认为图的相对边界是相邻的，从而可形成块；也就是，最高和最低行及第 1 列和最后 1 列可以形成块。

5. 首先寻找最大的块来获得最好的化简。

将这些原则应用于 F 卡诺图中产生了图 3-8所示的圆圈。



F(W,X,Y,Z)

图 3-8

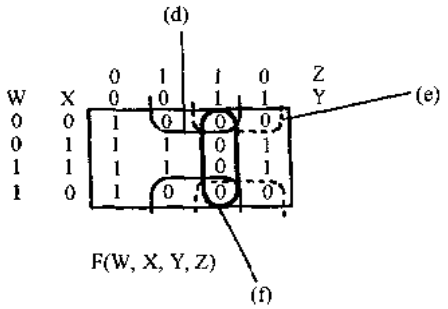
现在通过观察得到简化的布尔表达式。每个含 1 的块(包围圈)代表 1 项；因此会产生含有变量的 3 个组，这些变量可以很容易地用它们相应块的坐标确定下来。例如，限定在第 1 列的包围圈(a)对应于 Y = 0 和 Z = 0。因为 W 和 X 包含两个可能的值，利用结合律将其忽略。以相同的方式获得剩下的两项产生以下方程式，其中对应每一项的块由与图 3-8 相关的字母表明：

$$F = \underbrace{Y'Z'}_{(a)} + \underbrace{XY'}_{(b)} + \underbrace{XZ'}_{(c)}$$

以上描述的方法称作包围 1。回顾前面的内容，通过写出真值表包含输出为 0 的项的表达式，就可以用布尔代数创建函数的逻辑反的方程式。这种情况下，在卡诺图中用包围 0 的块替代包围 1 的块，然后像以前一样接着进行就能够得到函数的简化。称这种方法为包围 0，本例中，将产生简化的 F′而不是 F 的表达式。

例 3.6　在图 3-7(c)的卡诺图中用包围的 0 方法获得简化的 F′的函数。

见图 3-9。



$$F' = \underbrace{X'Z}_{(d)} + \underbrace{X'Y}_{(e)} + \underbrace{YZ}_{(f)}$$

图 3-9

3.5　卡诺图的应用

无关项

在数字逻辑的许多应用中，经常有永远不会发生的独立变量的组合。具有 BCD 输入的系统就是这样一种情况：BCD 数总有 4 位二进制数，相当于十进制数 0 到 9，因为 4 位数有 16 种惟一的组合，因此有 6 种（表示为十进制数 10 到 15）不使用的组合。代表不会发生的输入组合的布尔项称作“无关项”，因为设计者不关心这些项的输出是什么。

考虑接收 BCD 输入产生的输出用来驱动 7 段显示的接口电路的逻辑实现（这个块的真值表在第 2.2 题中论述）。段 a 的说明画在图 3-10 的卡诺图中，其中将 BCD 位标记为 W，X，Y，Z。

W	X	Z=0,Y=0	Z=1,Y=0	Z=1,Y=1	Z=0,Y=1
0	0	1	0	1	1
0	1	0	1	1	1
1	1	X	X	X	X
1	0	1	1	X	X

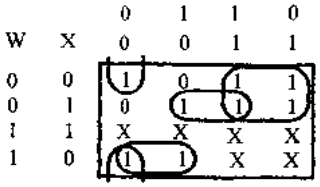
X 表明“无关”项

“a”的卡诺图

图 3-10

惯用的包围 1 的方法在图 3-11 中产生段 a 的卡诺图，得到的结果为

$$a = W'Y + X'Y'Z' + W'XZ + WX'Y'$$



“a”的卡诺图

图 3-11

因为包含 X 的输入组合从来不会出现，我们可以随意给 X 分配任意值，很明显使 X = 1 是有利的，因为这创建了扩充卡诺图包围圈的额外的机会。包围 1 的方法利用无关项，简化的布尔方程式，如图 3-12 所示。

很明显因为这个方程式比没有使用“无关项”获得的方程简单，所以看来在 3.4 节中我们在简化原则的列表中添加一条另外的覆盖原则是合理的：

6. 可用无关项来扩充块的覆盖，而不需包围它们自己。

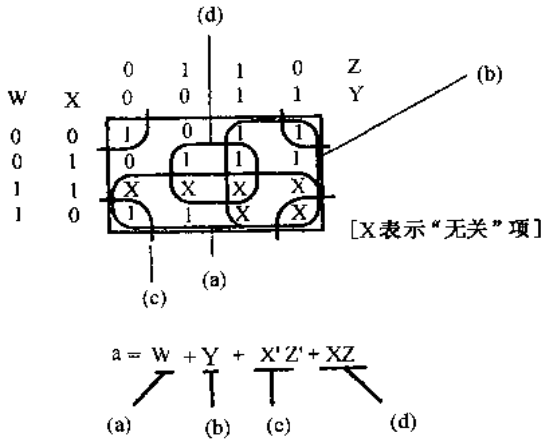


图 3-12

提示：没有必要或根本不用包含所有的无关项（Xs），这一点是重要的。根据设计者的使

用需要可将其取 1 或 0。

5 和 6 变量卡诺图

随着逻辑表达式中变量数目增加超过了 4 个，卡诺图的图形识别的好处开始降低。然而设计者可以利用对称映射扩充卡诺图的应用。考虑图 3-13 中所示的 5 个变量卡诺图，注意坐标轴仍是格雷编码，这保证了相邻的方框(映射位置)仅有 1 个变量不同，然而这并不表明所有的仅有 1 个变量之差的方框是物理上相邻的。通过创建位于 3 个变量坐标中间的反射轴来确定与简化过程相关的非物理相邻项的位置；关于轴镜像对称的这些方框意味着仅有 1 个变量之差。

			0	1	1	0	E
A	B	C	0	0	1	1	D
0	0	0	0	0	0	0	
0	0	1	W	X	0	0	
0	1	1	W	0	Y	Y	
0	1	0	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	Y	Y	
1	0	1	0	X	0	0	
1	0	0	0	0	0	0	

反射轴

图 3-13

WW 项在通常意义上讲是相邻的，并且产生了简化项 A′CD′E′。XX 项关于反射轴对称，通过观察表明它们仅有一个变量 A 之差，因此简化项是 B′CD′E。当考虑到反射轴时 YYYY 项形成含 4 个方框的块，就产生了简化项 BCD。

对于 6 个和更多变量扩充放射的概念是可能的，但图形的识别变得更加困难 。

例 3.7　具有两个反射轴的 6 变量卡诺图如图 3-14 所示。XXXX 项关于两个轴对称，对应于 B′CE′F。8 个 Y 项也关于两个轴对称，对应于 BCE。两个 W 仅仅关于垂直轴对称形成了稍微简化的项 AB′C′EF。

			0	1	1	0	0	1	1	0	F
			0	0	1	1	1	1	0	0	E
A	B	C	0	0	0	0	1	1	1	1	D
0	0	0	0	0	0	0	0	0	0	0	
0	0	1	0	X	0	0	0	0	X	0	
0	1	1	0	0	Y	Y	Y	Y	0	0	
0	1	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
1	1	1	0	0	Y	Y	Y	Y	0	0	
1	0	1	0	X	0	0	0	0	X	0	
1	0	0	0	0	W	0	0	W	0	0	

图 3-14

6 变量的卡诺图

6 个变量卡诺图的另一种方法就是创建 4 变量的卡诺图，并且在心里认为它们是向上逐层堆叠的(见例 3.8)。如果这些思想实现了，那么可以用同样方式使用相邻的基本规则

例 3.8　6 变量卡诺图的另一种方法如图 3-15 所示，在以 EF = 00，EF = .01，EF = 10，EF = 11 的顺序叠放这 4 张小卡诺图之后，我们发现以下的相邻项：

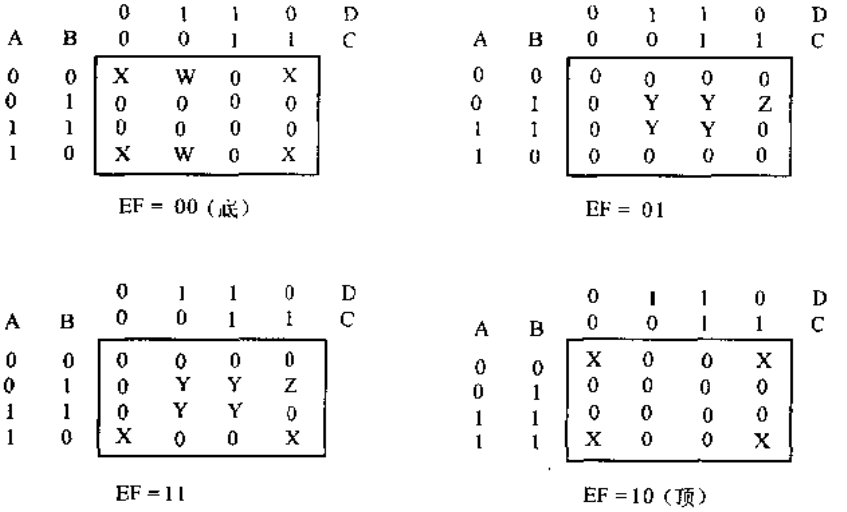


图 3-15

WW	仅仅在最底的卡诺图上相邻(产生 B′C′DE′F′项)。
YYYYYYYY	在两图内和在两图之间相邻(产生 BDF 项)。
XXXXXXXX	在最上层和最下层内和之间相邻(产生 B′D′F′)。 提示：在 EF = 11 图上的两个 X 项没有包含在这组中。
ZZ	在中间两图之间相邻(产生了 A′BCD′F)。

计算机辅助化简

正如我们可以从前面的例子中看出，随着变量数目增至 5 个或更多，识别图中相邻方格就变得更加繁重。幸运的是有相当多的软件可用来简化布尔代数，使我们能用计算机来获得多变量表达式的简化。通常这个软件具有多种能力，采用智能性的引导尝试和检错的算法。

画出异或函数的卡诺图

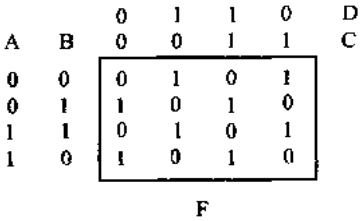


图 3-16

异或函数趋向于画成棋盘的形状。考虑第 2.10 题中如图 3-16 所示的 布尔表达式 F = (A⊕B)⊕(C⊕D)的卡诺图。因为没有相邻项，我们得到的结论是所给定的函数就是所能表达的最简形式。

很多情况下，如果将图处理成棋盘或类似棋盘的形式，使用异或可以非常有效地简化相关函数。

例 3.9　假设卡诺图如图 3-17 所示，我们希望得到最优

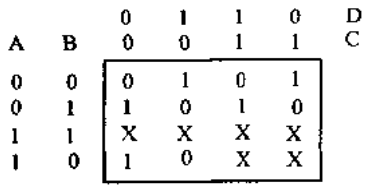
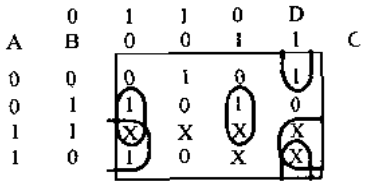


图 3-17



F = AD' + A'B'C'D + B'CD' + BCD + BC'D'

图 3-18

的简化形式。恰当地分配0和1，利用最后两行的无关项可以创建相应于图3-16卡诺图的函数的棋盘形式。将这个结果与图3-18中所示的通过相邻化简得到的表达式相比较是非常有趣的。

习题解答

3.1 说明 $A + A' = 1$

解 因为逻辑变量A只能是1或0，如果A=0，那么A′=1。反之亦然。因此，给定表达式的左边永远也不会有两个0，并且利用基本定律1这个结论是显然成立的。

3.2 证明摩根定律的两个形式。

解 借助穷举证明。

1. $AB = (A' + B')'$ 正如表3.2中所看到的，对于变量A和B的每一个组合，两个框起来的列是相等的。

表3.2

A	B	AB	A′	B′	A′+B′	(A′+B′)′
0	0	0	1	1	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	0	1

2. A+B=(A′B′)。见表3.3。两个框起来的列又一次证明了这个定律。

表3.3

A	B	A+B	A′	B′	A′B′	(A′B′)′
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

3.3 用穷举证明以下的布尔等式：$AB + BC + A'C = AB + A'C$。

解 给每一个变量和项用列建立真值表(表3.4)。如果对于每个独立变量的组合，给定方程式的左边和右边有同样的逻辑值。那么这两个表达式是相等的。

表3.4

A	B	C	A′	AB	BC	A′C	AB+A′C+BC	AB+A′C
0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	1	1	1
0	1	0	1	0	0	0	0	0
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	1	1
1	1	1	0	1	1	0	1	1

相同

证明给定的等式的另一种(更简单的)方法是利用卡诺图化简(参考题 3.7)。

3.4 证明两个给定的布尔表达式分别与异或函数及其取反相等,如式(3.15)和式(3.16)所表明的。

$$AB' + BA' = A \oplus B \tag{3.15}$$

$$AB + A'B' = \overline{A \oplus B} \tag{3.16}$$

解 通过观察 A⊕B 的真值表可以很容易证明式(3.15)是正确的。利用摩根定律可以证明式(3.16)[式(3.12a)和式(3.12b)]:

$$\overline{A \oplus B} = \overline{AB' + BA'} = (A' + B)(B' + A) = A'B' + A'A + BB' + BA$$

因为 A′A 和 BB′都是 0,这样就剩下了式(3.16)所表达的等式。

3.5 证明或反证以下等式。如果它被证明了,它遵循 ACD=0 吗?

$$A'C + CD + AB'C' = A'C + CD + AB'C' + ACD$$

解 建立起对每个项的列的真值表,如表 3.5 所示。LS 代表左边(等式的),RS 代表右边。因为独立变量的所有可能组合都有 LS= RS,所以通过穷举证明了这个等式。提示:这并不表明 ACD 为 0。然而,这一项是多余项,因为当剩下的其中一项也为 1 时它才为 1。

表 3.5

A	B	C	D	A′C	CD	AB′C′	ACD	LS	RS
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1	1
0	0	1	1	1	1	0	0	1	1
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0	1	1
0	1	1	1	1	1	0	0	1	1
1	0	0	0	0	0	1	0	1	1
1	0	0	1	0	0	1	0	1	1
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	1	0	1	1	1
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	1	0	1	1	1

3.6 证明以下布尔等式。

(a) $A(B' + C) = AB' + AC$

(b) $A(A + B'C) = A$

(c) $A'B' + AB' + AB = A + B'$

解 (a) $A(B' + C) = AB' + AC$。运用穷举得到表 3.6。

表 3.6

A	B	C	B′	(B′ + C)	A(B′ + C)	AB′	AC	AB′ + AC
0	0	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
1	0	0	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0
1	1	1	0	1	1	0	1	1

因为在表 3.6 中所表出的黑体项对于独立变量 A, B 和 C 的所有可能的组合都是相配的，等式被证明。

(b)　A(A + B′C) = A。运用穷举可以得到表 3.7。

表 3.7

A	B	C	B′C	A + B′C	A(A + B C)
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	0	1	1

运用代数运算：

$$A(A + B'C) = A + AB'C \qquad \text{(分配律和定律 5)}$$
$$= A(1 + B'C) \qquad \text{(分配律)}$$
$$= A \qquad \text{(定律 1 和 3)}$$

(c)　A′B′ + AB′ + AB = A + B′。运用穷举得到表 3.8.

表 3.8

A	B	C	A′	B′	A′B′	AB′	AB	LS	RS
0	0	0	1	1	1	0	0	1	1
0	0	1	1	1	1	0	0	1	1
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	1	0	1	1
1	0	1	0	1	0	1	0	1	1
1	1	0	0	0	0	0	1	1	1
1	1	1	0	0	0	0	1	1	1

运用代数运算：

$$A'B' + AB' + AB = (A' + A)B' + AB \qquad \text{(分配律)}$$
$$= B' + AB \qquad \text{(定律 6)}$$
$$= B' + A \qquad \text{[方程(3.14b)]}$$

3.7　用卡诺图证明(AB + BC + A′C) = (AB + A′C)。

解　☞ 表达式的左边的卡诺图如图 3-19 所示。

第3行的第一个1用坐标ABC′定义，相邻的1用ABC定义。因为ABC′ + ABC = AB(C + C′) = AB，所以第3行中的两个1定义为AB项。因为变量C有两个可能的值，所以被删去。(参考第3.4节讨论的内容)

同样，BC可以画出第2列的两个1并确定位置，因此A在一种情况下为0，另一种情况下为1。同样A′C也位于第2列，并且这个位置可以删去B。现在我们可以画出邻近1的包围圈来得到函数的简化，如图3-20所示。

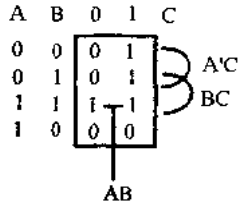


图3-19

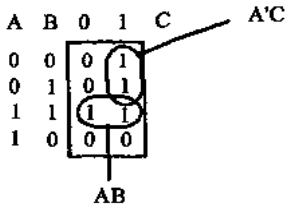


图3-20

所有的1用图示的两个区域考虑到了，因此函数可以表示为AB + A′C。我们看到BC项是多余的，如第3.3题的说明。

3.8 画出函数F(A, B, C, D) = AB′C + B′D′ + BCD′ + AB′C′D。的卡诺图。

解 ☞ 参考图3-21。AB′C是第4行(AB′)与第3、4列(D)的交叉。B′D′是第1、第4列(D′)和第1、第4行(B′)的交叉。BCD′是第2、第3行(B)和第4列(CD′)的交叉。AB′C′D是第4行(AB′)和第2列(C′D)的交叉。

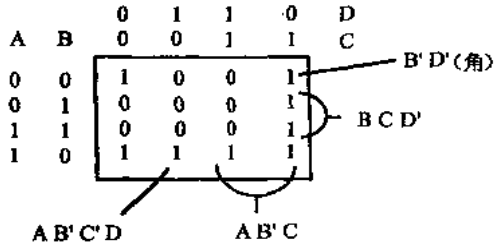


图3-21

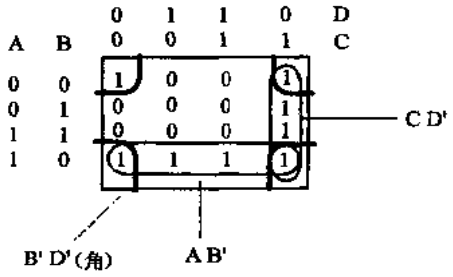


图3-22

3.9 利用第3.8题画出的卡诺图求出F的另一种简化形式。

解 ☞ 参考图3-22。CD′是由第4列组成的4×1矩形块。AB′是由第4行组成的1×4的矩形块。B′D′由4个角定义，也就是第1、第4行和第1、第4列的交叉。函数是

$$F = CD' + AB' + B'D'$$

3.10 用第3.8题画出的卡诺图求出F′的简化表达式。

解 ☞ 参考图3-23。前面所示的包围0的方法产生了需要的表达式。为了便于识别，每个项都在图上标出。

3.11 4个变量中当有3个变量为真时函数为真，画出该函数的卡诺图，并尽你所能写出最简的布尔表达式。

解 ☞ 用A, B, C, D作为输入变量画出卡诺图的轴，数出每个方格中坐标轴1的数目，见图3-24。我们所能做的最好的化简是包含两个1的4个区域，每个都在图3-25中标明。这就产生了函数：

$$\begin{aligned} F &= BCD + ABC + ACD + ABD \\ &= AB(C + D) + CD(A - B) \end{aligned}$$

3.12 对具有BCD输入的7段显示驱动器的输出b段利用无关项获得简化的布尔表达式。参考第2.2题。

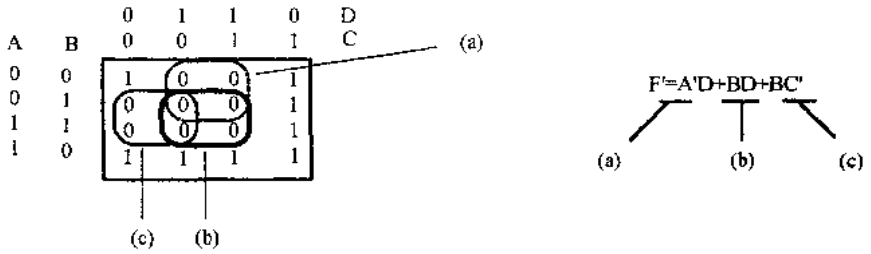


图 3-23

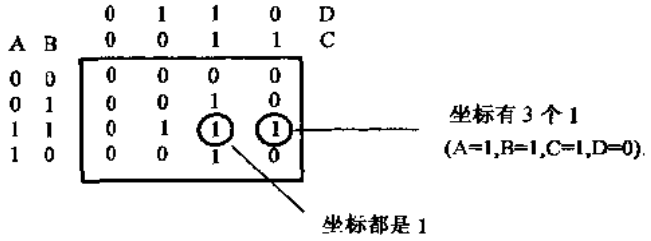


图 3-24

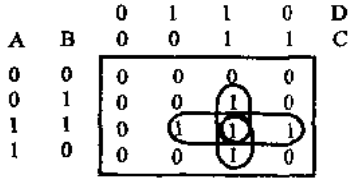


图 3-25

解 ☞ 卡诺图由第 2.2 题的真值表推导出，其中 W, X, Y, Z 代表给定的 BCD 数码的 4 位。见图 3-26。

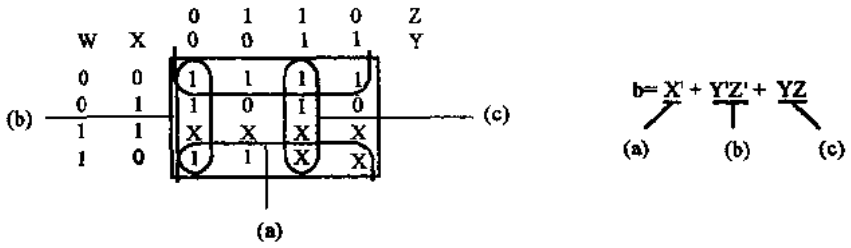


图 3-26

3.13 7 段显示电路的 BCD 码的输出 b 的布尔表达式包含项 Y′Z′ + YZ。参考 3.5 节，可以看到 a 段的表达式也包含这样的结构。证明这个项可以通过异或的逻辑取反实现，而不是使用 3 个门。

解 ☞ 画出 F = Y′Z′ + YZ 的真值表，如表 3.9 所示。

表 3.9

Y	Z	F	F′
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

提示：F′是异或函数(见 2.3 节)，因此它的非就是所要求的 F 函数。所以 b 段逻辑图可以用图 3-27所示的简化形式画出。

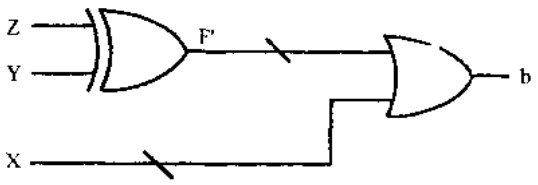


图 3-27

A	B	C	0 0	0 1	1 0	1 1	E D
0	0	0	0	0	0	1	
0	0	1	1	1	0	0	
0	1	1	0	0	1	1	
0	1	0	1	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	1	1	
1	0	1	0	0	0	0	
1	0	0	0	0	0	1	

图 3-28

3.14　如图 3-28 所示为 5 变量函数的卡诺图，求出 F(A, B, C, D, E)的最简表达式。

解　利用反射轴和包围 1 的方法能够获得以下的函数，它的项在图 3-29 所示的卡诺图上标出。

$$F = \underline{B'C'DE'} + \underline{A'B'CD'} + \underline{A'BC'D'E'} + \underline{BCD}$$

(c)　(a)　(d)　(b)

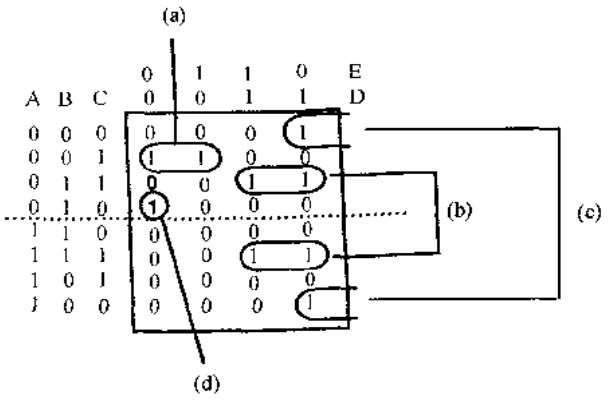


图 3-29

A	B	C	0 0	0 1	1 1	1 0	E D
0	0	0	X	0	0	1	
0	0	1	1	1	0	0	
0	1	1	X	X	1	1	
0	1	0	1	0	0	X	
1	1	0	0	0	0	0	
1	1	1	X	X	1	1	
1	0	1	0	0	0	0	
1	0	0	X	X	0	1	

F

图 3-30

3.15　图 3-30 给出了 5 变量的包含一些无关项的卡诺图，用包围 0 和包围 1 的方法将其化简。

解　参考图 3-31，包含 1 的方法可以得到下式：

$$F = \underset{(c)}{BC} + \underset{(d)}{A'CD'} + \underset{(b)}{B'C'E'} + \underset{(a)}{A'C'E'}$$

提示：可以独立处理每一半卡诺图；也就是包围不需要关于反射轴对称[例如(a)和(d)]。

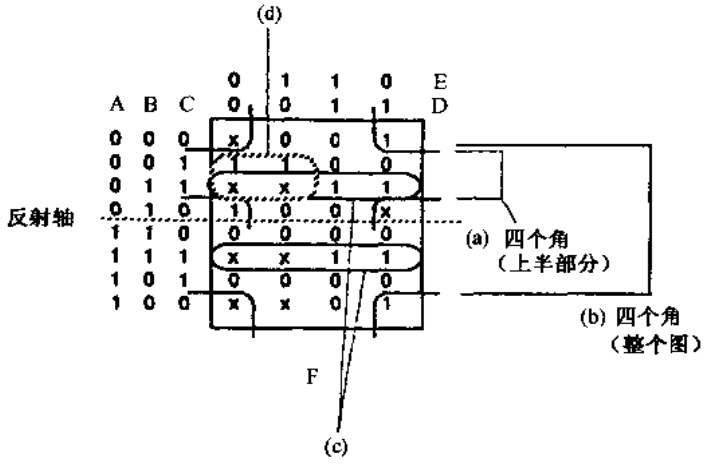


图 3-31

包围 0 的方法如图 3-32 所示。因为对称性，可将包围区域(a)和(c)看作是 8 个 0 的单独的块，这产生了 C′E 项。为了证实这一点，对它们分开考虑：

(a)BC′E　　(c)B′C′E

$$BC'E + B'C'E = C'E(B + B') = C'E$$

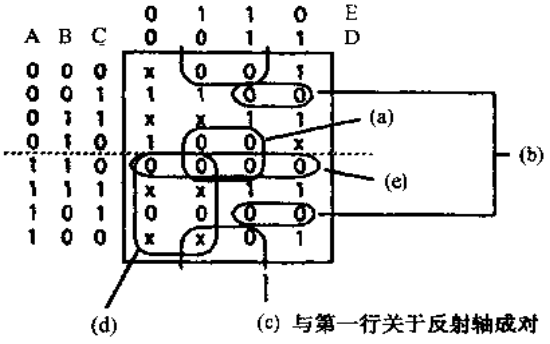


图 3-32

因此，将所求的布尔表达式看作是

$$F' = \underset{(d)}{AD'} + \underset{(b)}{B'CD} + \underset{(a),(c)}{C'E} + \underset{(e)}{ABC'}$$

3.16　设计具有如表 3.10 所示的真值表的数字组合逻辑系统。输入是 A, B, C 和 D，输出为 F。无关项对应于永远不会出现的输入组合，因此是任意状态。

(a)画出卡诺图，尽你所能化简并写出 F 的布尔表达式。

(b)画出对应于简化函数 F 的逻辑图。

表 3.10

	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	x
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	x
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	x
11	1	0	1	1	1
12	1	1	0	0	x
13	1	1	0	1	x
14	1	1	1	0	x
15	1	1	1	1	x

解 (a)可直接从真值表中读出每个 1 和每个无关项的卡诺图的坐标。例如:第 6 行告诉我们 X 应该放在 A=0, B=1, C=1, D=0 的位置,它位于图 3-33 所示的卡诺图的第 2 行和第 4 列的交叉处。通过简化,我们得到图 3-34、结果方程式为

$$F = C + \underbrace{B'D'}_{(a)}$$
(b)

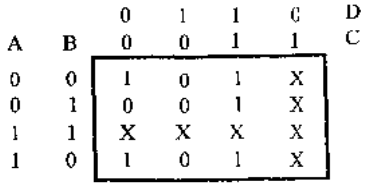


图 3-33

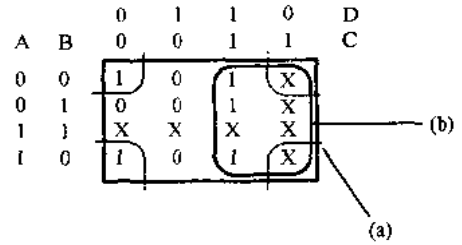


图 3-34

(b)通过观察画出逻辑图(见图 3-35)

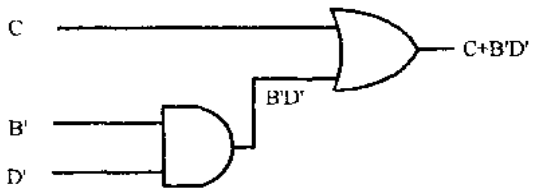


图 3-35

3.17 图 3-36 中给出逻辑图,

(a) 写出 F 的逻辑方程式。

(b) 尽可能化简逻辑。

(c) 画出化简后的逻辑图。

$$F = \underbrace{XYZ'}_{1} + \underbrace{W'YZ}_{2} + \underbrace{XYZ}_{3} + \underbrace{WXZ.}_{4}$$

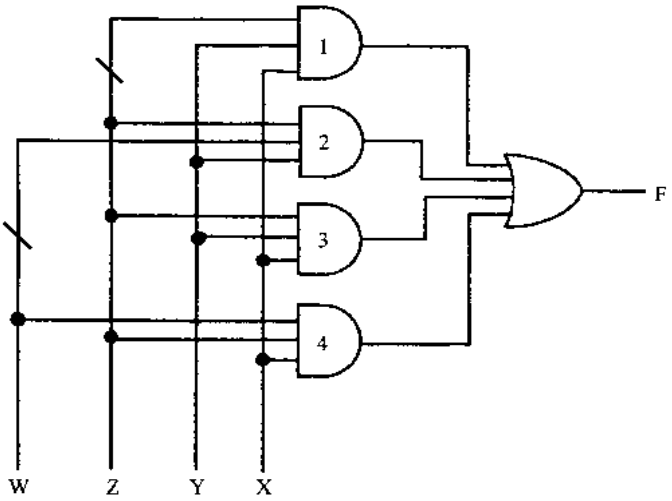


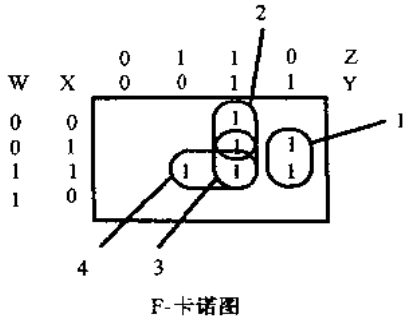
图 3-36

解　(a) 用数字 1 到 4 标出与每一项相关的与门。

(b) 将以上方程中每一项都作为两个 1 画在卡诺图中，如图 3-37 所示。

很明显，将第 1 项和第 3 项合并，在一个包围区围绕起四个 1 可完成化简，如图 3-38 所示。方程式结果为

$$F = \underbrace{XY}_{(a)} + \underbrace{W'YZ}_{(b)} + \underbrace{WXZ}_{(c)}$$



F-卡诺图

图 3-37

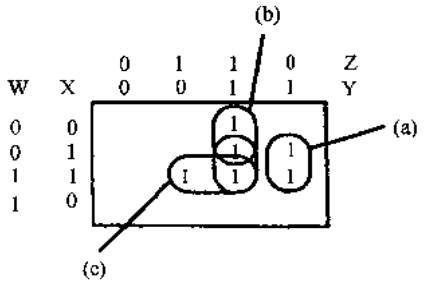


图 3-38

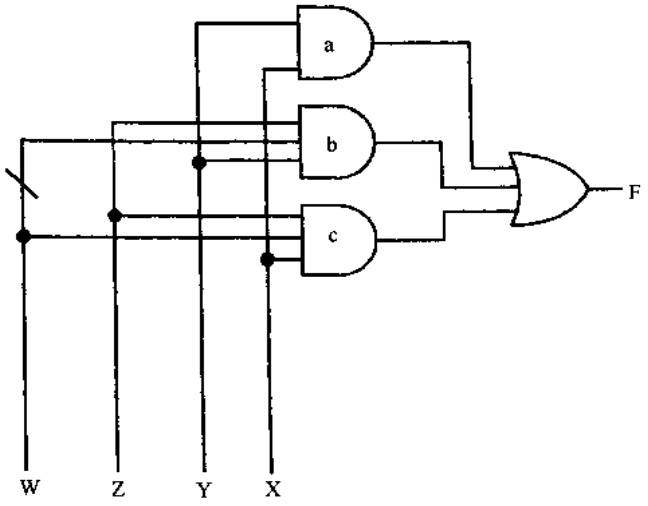


图 3-39

(c) 上面标出的 3 项对应于图 3-39 的逻辑图中与门的输出。

3.18 设计奇偶校验的逻辑电路。图示采用异或门完成这一目的。

解 奇偶校验要确定在给定的二进制字中 1 的数目为奇数还是偶数。不用奇校验一定暗指二进制数 0 或者偶校验;不会存在其他情况。

考虑图 3-40 所示的异或电路。当 A 和 B 这两个输入不同时,相当于 G_1 门的输入有奇数个 1 时,A+B 结果为 1。下一步,考虑 G_2。当在它的输入端有奇数个 1 时,输出 F 为 1,意味着当 C=0 时,G_1 一定产生 1,或者如果 G_1 为 0,C 一定是 1。这存在两种可能:

1. 如果 $G_1=1$,那么 A,B 是奇数(有奇数个 1),并且如果 C=0,那么 F 只能为 1。
2. 如果 $G_1=0$,那么 A,B 是偶数或 0,并且如果 C=1,那么 F 只能为 1。

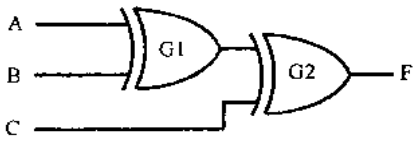


图 3-40

在这两种情况下,只有当 A,B,C 有奇数个 1 时(奇校验),F 能够为 1。这通过构建 F 的卡诺图可以很清楚地说明,如图 3-41 所示。考虑增加第 3 个异或门的效果。

A	B	0	1	C
0	0	0	1	
0	1	1	0	
1	1	0	1	
1	0	1	0	

F

图 3-41

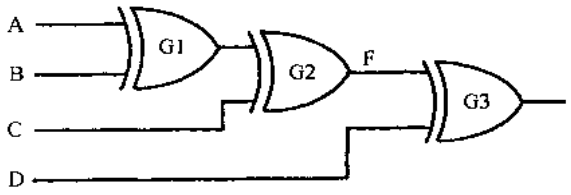


图 3-42

如图 3-41 所示,当 A,B,C 为奇数时 F 为 1。如果 F=1 并且 D=0[(A,B,C,D)为奇数]或者 F=0 并且 D=1 时,G_3 将产生 1。后者的情况下,A,B,C 一定是偶数或者是 0,并且 D=1 的必要性要求 A,B,C,D 具有奇校验。应该清楚,电路可以通过无限制地级连或门进行扩充,因此可以检验任何长度二进制数的奇偶性。

3.19 当第 3.18 题的异或电路的输入有奇校验时它会产生 1。说明对于这种输出的逻辑取反会因为偶校验输入产生 1。

解 输出端逻辑取反意味着对于每一个输入变量组合,输出由 1 变为 0,0 变为 1。结果的卡诺图如图 3-43 所示。一眼就可以看出当 A,B,C 是 0 或偶数时这种安排将产生 1,并且相应的布尔表达式表示取反的异或函数 $F'=A\oplus B\oplus C$。(参考 3-4 题)

如果我们有能够对输出取反的硬件,仅仅需要将其添加在第 3.18 题的电路中来将 1 为奇校验转换成 1 为偶校验。逻辑取反的硬件实现将在第 4.3 节讨论。

A	B	0	1	C
0	0	1	0	
0	1	0	1	
1	1	1	0	
1	0	0	1	

F'

图 3-43

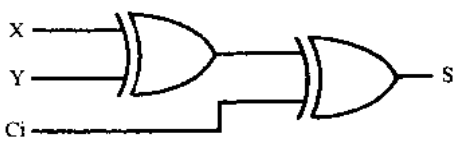


图 3-44

3.20 说明可用异或门实现 3.4 节中的 1 位加法器和的输出。

解 S的卡诺图与第3.18题3变量的异或函数相同，这意味着两个函数有同样的真值表并且一定相等。因此，和输出的实现如图3-44所示。

3.21 利用题2.16中定义的方框，求出由图3-45所示系统完成的简化逻辑表达式。

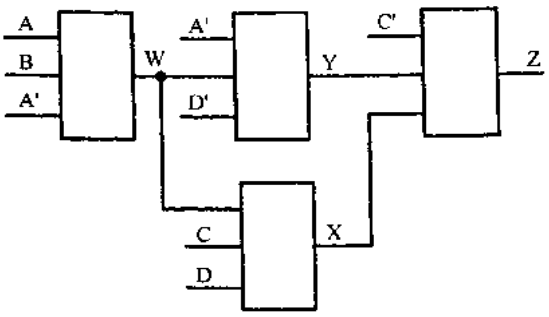


图3-45

解 利用题2.16得到的真值表创建正确的卡诺图，产生了图3-46。进行所标变量的替代后，利用布尔代数化简框图输出的结果的逻辑表达式：

A	B	0	1	C
0	0	0	0	
0	1	0	1	
1	1	1	1	
1	0	0	1	

F

$F = AB + BC + AC$

图3-46

$$W = AB + A'B + AA' = AB + A'B + 0 = B(A + A')$$
$$= B \cdot 1 = B$$
$$X = CD + CW + DW$$
$$= CD + BC + BD$$
$$Y = A'W + A'D' + D'W$$
$$= A'B + A'D' + BD'$$
$$Z = XY + XC' + YC'$$
$$= (CD + BC + BD)(A'B + A'D' + BD') + (CD + BC + BD)C' + (A'B + A'D' + BD')C'$$
$$Z = T_1 + T_2 + T_3$$

这里

$$T_1 = (CD + BC + BD)(A'B + A'D' + BD')$$
$$= A'BCD + A'BC + A'BD + A'BCD' + BCD'$$
$$T_2 = (CD + BC + BD)C' = BC'D$$
$$T_3 = (A'B + A'D' + BD')C' = A'BC' + A'C'D' + BC'D'$$

利用卡诺图可以简化Z的表达式。从T_1开始，如图3-47所示。将T_2项和T_3项插入到T_1卡诺图中，产生了如图3-48所示的组合卡诺图，从中可以得到所要求的函数：

3.22 利用第2.2题求出的7段显示驱动器的真值表，用卡诺图来化简逻辑。

解 卡诺图如图3-49所示。

Z = A'C'D'+ A'B + BD' + BC'

A	B	0 0	1 0	1 1	0 1	D C
0	0	0	0	0	0	
0	1	0	1	1	1	
1	1	0	0	0	1	
1	0	0	0	0	0	

T1

T1 = BCD' + A'BD

图 3-47

A	B	0 0	1 0	1 1	0 1	D C
0	0	1	0	0	0	
0	1	1	1	1	1	
1	1	1	1	0	1	
1	0	0	0	0	0	

Z

图 3-48

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	0	1	1	
0 1	0	1	1	1	
1 1	x	x	x	x	
1 0	1	1	x	x	

"a"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	1	1	1	
0 1	1	0	1	0	
1 1	x	x	x	x	
1 0	1	1	x	x	

"b"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	1	1	0	
0 1	1	1	1	1	
1 1	x	x	x	x	
1 0	1	1	x	x	

"c"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	0	1	1	
0 1	0	1	0	1	
1 1	x	x	x	x	
1 0	1	1	x	x	

"d"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	0	0	1	
0 1	0	0	0	1	
1 1	x	x	x	x	
1 0	1	0	x	x	

"e"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	1	0	0	0	
0 1	1	1	0	1	
1 1	x	x	x	x	
1 0	1	1	x	x	

"f"

W X	0 0	0 1	1 1	0 1	Z Y
0 0	0	0	1	1	
0 1	1	1	0	1	
1 1	x	x	x	x	
1 0	1	1	x	x	

"g"

图 3-49

一组解答为

a = W + Y + XZ + X′Z′

b = X′ + YZ + Y′Z′

c = X + Y′ + Z

d = W + XY′Z + X′Y + X′Z′ + YZ′

e = X′Z′ + YZ′

f = W + Y′Z′ + XY′ + XZ′

g = W + YZ′ + X′Y + XY′

因为相关的大量的无关项，就存在许多可能的排列。所选的这组特别的答案利用了多次出现的单项，这对于减少逻辑门的数量是有益的。在获得优化的化简时经验往往是一个重要因素。

补充习题

3.23 化简以下布尔函数以减少变量和项的数目。

(a) $BCD + B'C + BCD'$

(b) $(A + B)'(A' + B')'$

(c) $(A + AB' + CD)'$

(d) $B'(B + C) + D' + CD$

(e) $\{A[B + C(A' + B)]\}'$

3.24 求出下面每一个布尔函数的补并尽可能将其化简。

(a) $[(AA' + B)']B$

(b) $(AB + AC')$

(c) $A'BC + ACD + A'BC' + B'D$

(d) $A'B' + DC'$

3.25 用卡诺图化简题2.20描述的乘法器的设计方程并确定所需的逻辑门。

3.26 在题2.24中，得到了简单海明编码的奇偶校验位的布尔方程式。求出 P_3 的异或实现。

$$P_3 = D'_4D'_3D'_2D'_1 + D'_4D'_3D'_2D_1 + D'_4D_3D_2D'_1 + D'_4D_3D_2D_1 + D_4D'_3D_2D'_1 + D_4D'_3D_2D_1 + D_4D_3D'_2D'_1 + D_4D_3D'_2D_1$$

3.27 在题2.25中，研究了亮灯问题的真值表。画出卡诺图和逻辑图。

3.28 化简题2.28的减法器的布尔方程。

3.29 利用图3-50中的卡诺图，尽你所能利用"无关项"来简化F的表达式。要求数量最小但包围圈最大。

		0	1	1	0	Z
W	X	0	0	1	1	Y
0	0	1	0	1	1	
0	1	0	1	1	1	
1	1	x	x	x	0	
1	0	1	0	x	x	

"F"卡诺图

图3-50

3.30 利用题2.33求出的真值表，画出输出 A_n 和 C_n 的卡诺图并且尽你所能利用"无关项"来简化F的表达式。要求数量最小但包围圈最大。

3.31 要求设计组合逻辑电路，在4位二进制输入相应的十进制数时，若被3整除就会亮，可知数字0，1，7，11，14作为输入时永远不会使灯亮。画出这个函数的卡诺图，用卡诺图包围1的方法尽量化简，并画出逻辑图。

3.32 有4个输入的系统，当且仅当输入为真的数目为奇数时，输出为真。设计出满足要求的逻辑图。

3.33 函数Z的6变量卡诺图如图3-51所示。用数量最小但包围圈最大来获得简化的Z的布尔表达式。

3.34 创建将4位二进制数转化为格雷码的转换器的卡诺图。将二进制位记为D，C，B，A，而格雷码位记为W，X，Y，Z。

3.35 求出题3.34中将4位二进制数转化为格雷码转换器的简化布尔方程式。

3.36 用卡诺图证明：$A'B + C'D' + A'B'C' = C'A' + A'BC + AC'D'$。

3.37 证明 $A'B + C'D' + A'B'C' = (B'C + AC + AD)'$

3.38 用摩根定律和其他布尔代数求出 $F = (B'C + AC + AD)'$ 的另一种表达形式。

3.39 一设备当且仅当3输入中大多数为真时输出为真，列出该设备的真值表，写出布尔方程式并将其简化。

			0	1	1	0	0	1	1	0	F
			0	0	1	1	1	1	0	0	E
A	B	C	0	0	0	0	1	1	1	1	D
0	0	0	0	1	0	0	0	0	0	0	
0	0	1	0	1	0	0	0	0	1	0	
0	1	1	0	0	1	1	X	1	0	0	
0	1	0	0	0	0	X	X	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
1	1	1	0	0	1	1	1	X	X	0	
1	0	1	0	1	0	0	0	0	1	0	
1	0	0	0	X	1	0	0	1	0	0	

图 3-51

补充习题答案

3.26

		0	1	1	0	D1
D4	D3	0	0	1	1	D2
0	0	1	1	0	0	
0	1	0	0	1	1	
1	1	1	1	0	0	
1	0	0	0	1	1	

P3

		0	1	1	0	B0
A1	A0	0	0	1	1	B1
0	0	0	0	0	0	
0	1	0	0	0	0	
1	1	0	0	1	0	
1	0	0	0	0	0	

C3=A1 A0 B1 B0（一个 4 输入与门）

		0	1	1	0	B0
A1	A0	0	0	1	1	B1
0	0	0	0	0	0	
0	1	0	0	0	0	
1	1	0	0	0	1	
1	0	0	0	1	1	

C2=A1 A0′B1+A1 B1 B0′
（两个 3 输入与门和一个 2 输入或门）

		0	1	1	0	B0
A1	A0	0	0	1	1	B1
0	0	0	0	0	0	
0	1	0	0	1	1	
1	1	0	1	0	1	
1	0	0	1	1	0	

C1=A1 B1′ B0+A1′ A0 B1+A0 B1 B0′+A1 A0′ B0
（4 个 3 输入与门和 1 个 4 输入或门）

A1	A0	0	1	1	0	B0
		0	0	1	1	B1
0	0	0	0	0	0	
0	1	0	1	1	0	
1	1	0	1	1	0	
1	0	0	0	0	0	

C0=A0 B0（1个2输入与门）

图 3-52

既然包围1的区域都不涉及 D_1，我们用3个变量重新画出函数的卡诺图。

D4	D3	0	1	D2
0	0	1	0	
0	1	0	1	
1	1	1	0	
1	0	0	1	

P3

当输入 D_4, D_3, D_2 有偶数个1时，然后1作为输入出现。因此，我们一定要利用题3.19中讨论的添加反向器的异或配置，如图3-53所示。一个附加的异或门上有一输入连在逻辑1并以此作为所需的反向器。

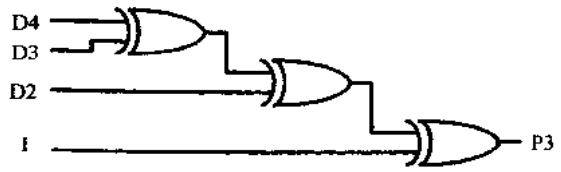


图 3-53

3.27　见图3-54。提示：棋盘形式。

A	0	1	B
0	1	0	
1	0	1	



图 3-54

3.28　$D = X \oplus Y \oplus B_i$　　$B_o = X'Y + X'B_i + YB_i$

3.29　因为布尔代数不是化简问题的运算，因此几个答案是可能的。以下是作者努力的最好结果。

W	X	0	1	1	0	Z
		0	0	1	1	Y
0	0	1	0	1	1	
0	1	0	1	1	1	
1	1	0	1	1	0	
1	0	1	0	0	1	

F=XZ+X′Z+W′Y

"F"卡诺图

图 3-55

3.30　见图 3-56。

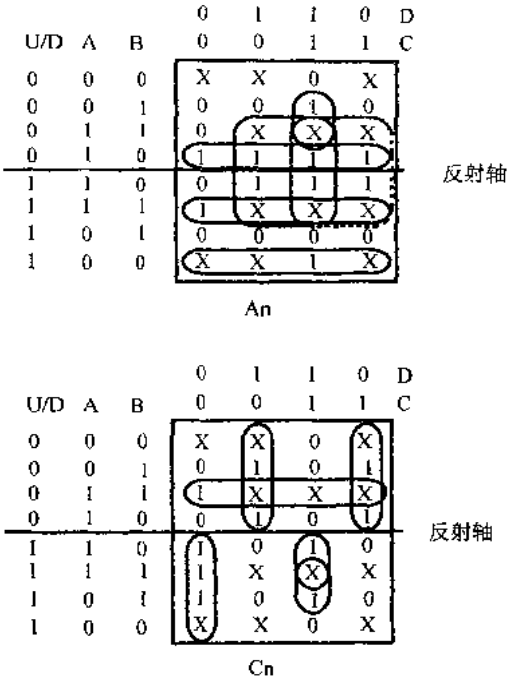


图 3-56

3.31　见图 3-57。

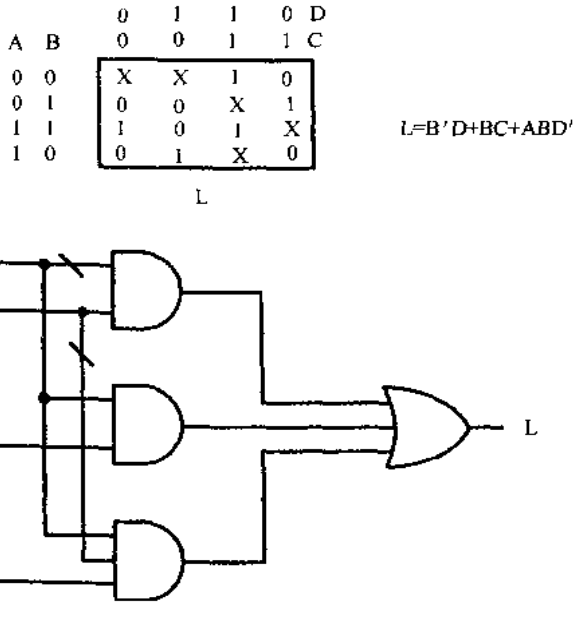


图 3-57

3.32　异或解答特别的精简(见图 3-58)。

3.33　Z = BCEF + BCE + AB′C′EF + B′CE′F + B′D′E′F

3.34　见图 3-59。

3.35

W = D

X = CD′ + C′D = C⊕D

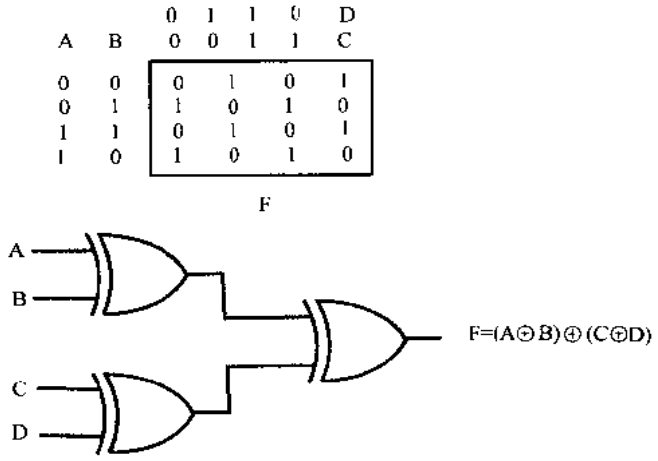


图 3-58

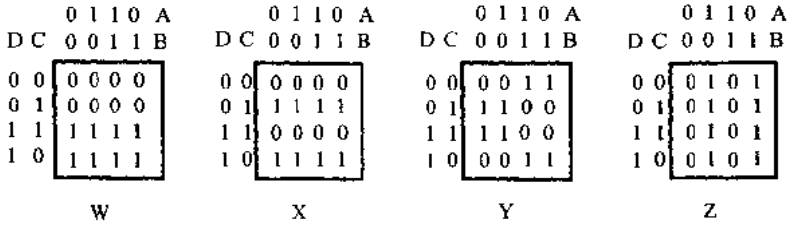


图 3-59

$$Y = BC' + B'C = B \oplus C$$

$$Z = AB' + A'B = A \oplus B$$

3.36　见图 3-60。

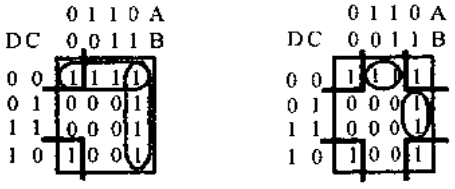


图 3-60

3.37　画出每个函数的卡诺图，左边是 F 包围 1 的方法(如图 3-60)。右边函数(括号中)是 F′的表达式，因此，用包含 0 的方法。两个卡诺图又是相同的，从而表明逻辑是相等的。

3.38　给出几种形式：

$$\begin{aligned}(B'C + AC + AD)' &= (B'C)'(AC)'(AD)' \\ &= (B + C')(A' + C')(A' + D') \\ &= (A'B + C')(A' + D') \\ &= A'C' + C'D' + A'B\end{aligned}$$

3.39　见表 3.11。

表 3.11

输　入			输　出
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$F = A'BC + ABC' + AB'C + ABC$(从真值表获得)；$F = AB + BC + AC$(化简形式)

第 4 章　硬件和混合逻辑规则

4.1　引言

组合逻辑设计及其硬件实现的关系，通常用约定的某种混合逻辑形式来描述，设计者选择好逻辑和电压值之间的关系，然后将表示逻辑和电气特性的符号综合在一起。混合逻辑既可作为将逻辑转换为硬件的设计工具(综合)，也可作为通过硬件来描述逻辑实现的手段(分析)，这两种应用在此都要讨论。

组合逻辑实现的基本硬件单元是门，它如同在 2.5 节所提到的是数字逻辑系统的基本模块。门电路(硬件)根据两个通常被称为高(H)和低(L)的电平进行操作，它们被称作逻辑 1 和逻辑 0(或者真和假)。设计者的任务就是在门信号和设计的逻辑值之间安排合适的比较对。

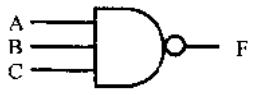
4.2　门电路硬件

门电路硬件以 IC 的形式存在，它在一片硅片上可集成几个到成百上千个门。具体门类型及其功能特性通常由生产商描述物理器件的电气特性、电压或电流、输入/输出(I/O)来说明。

例 4.1　硬件中最通用的类型之一就是所谓的与非门，它的 I/O 表如图 4-1 所示。虽然仅有 3 个输入，但是可产生具有各种输入数目的门。I/O 表列举了所有可能的输入电平组合及它们相应的输出电平。

输入			输出
A	B	C	F
L	L	L	H
L	L	H	H
L	H	L	H
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	L

电压 I/O 表



与非硬件符号

图 4-1

如果使图 4-1 的电平与逻辑值相等，那么可以将电压的 I/O 表解释为逻辑真值表(参考第 2.2 节)。因为在早期的逻辑设计中，通常指定逻辑值为最高电压相当于真(H=1 或 HT)。这种指定称作正向逻辑。

例 4.2　3 输入与非门硬件的正向逻辑图。指定 H = 1(同时意味着 L = 0)产生了图 4-2(a)所示的真值表。提示：这个真值表不满足逻辑与和逻辑或的定义，将其解释为一个与

输入			输出
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(a) 正向逻辑硬件真值表

输入			输出
A	B	C	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) 与真值表

输出
G'
1
1
1
1
1
1
1
0

(c) 与非

图 4-2

门紧跟着逻辑取反(逻辑与取反简化为与非)。图 4-1 中与非的符号用小圆圈表明取反。

4.3 将混合逻辑作为设计工具

给硬件任意分配逻辑值

许多设计者已经放弃了正向逻辑并开始使用所谓的混合逻辑概念。在这种方式下,给电压值分配逻辑值不是固定的,由设计者的喜好而定。

考虑例 4.1 的与非门的硬件混合逻辑图,其中指定输入 H = 1,而输出 F = 1。见表 4.1。

表 4.1

(HT)输入			(LT)输出
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

很明显真值表满足逻辑与函数的定义:也就是说当且仅当所有输入为真时输出为真。换句话说,如果做这样的分配,在输入端 L = 1(LT),那么在输出端 F = 1(HT),得到了当 1 个或多于 1 个输入为真时输出为真的真值表。

表 4.2

(LT)输入			(HT)输出
A	B	C	F
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

它被认为是逻辑或函数的描述。

提示:可使用相同的硬件来实现两种不同的逻辑符号。因此要给设计者提供相当的灵活度。

混合逻辑符号

为了避免混淆,对与非门硬件已经研究出了两种不同的逻辑符号。一个表明用作逻辑与,另一个表明用作逻辑或,它们在图 4-3 中给出。

用半箭头表示它出现在电路中的那一点是 LT。相反,没有半箭头表明 HT。许多设计者喜欢用小圆圈或小泡作为 LT 的标识符,而且经常是使用多余的双标,由此信号标上变量后缀 .L或.H表明与它们相联系的真值。这些替代形式如图 4-4* 所示。

和低-真信号同时使用的术语是低有效信号。在这种情况下,可将图 4-4 中的符号理解为

* 或非硬件的真值表分析和符号在问题解答第 4.1 题中给出。



用作或门的与非硬件　　用作与门的与非硬件

图 4-3



图 4-4　与非逻辑

带有低有效信号输入的或门或低有效信号输出的与门。

为了使混合逻辑工具在设计时变得容易，许多设计者用斜划线记号来表示逻辑取反。斜划线(最早在第 2.5 节中介绍过)是一种特殊的逻辑符号，我们将其理解为从变量到其逻辑取反的重要改变。提示：这个符号不代表硬件，它仅仅是逻辑元件。然而，它和半箭头的约定确实有关。如果将图 4-5 中的水平线认为是一条快速导通的电线(硬件)，那么因为导电特性，如果给线的左端提供任意给定的电压，右端一定是同样的电压。然而，假设指定 0V 代表逻辑假，那么更高一些的电压，比方说 5V 代表真。同时假设将 5V 电压加在线的左端，那么因为当左边是真时斜划线强行使我们将右端理解为假，这将会有明显的冲突，同时我们为自然规律所迫而希望两端都是高电压。这两个要求可以满足的惟一方法是将左端设计为 HT，而右端设计为 LT，或者相反。这表明每一个斜划线带有与其相关的作为逻辑和硬件桥梁的半箭头。(见图 4-6)

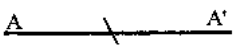


图 4-5

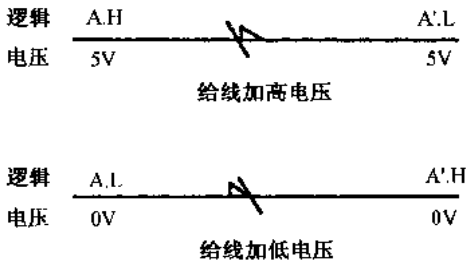


图 4-6

例 4.3　假设逻辑变量 A、B 和 C 都是 HT，希望画出函数 F = A′BC 的使用与非硬件的混合逻辑图。图 4-7 中的斜划线表明了在最终的硬件实现中创建 A′的重要性。

如果用图 4-7 表示硬件的话就存在一个问题。假设这个门由与逻辑实现。当在与非电路中实现时，当施加高输入时，它会产生低输出(认为是真)。输入 B 和 C 指定为 HT，满足输入要求，但是门的第 3 个输入不能为高，同时将其理解为 A′，因为斜划线标志要求 A′为 LT。当看图时注意到门输入和斜划线标志之间的线段包含不平衡箭头时，也就是在线段的一端有半箭头而另一端没

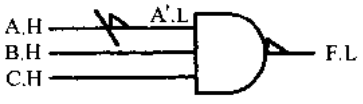


图 4-7

有，这个问题很容易识别。这表明虽然逻辑是正确的，但输入的真值是错误的。

电压反向器

真值问题可以由假定无逻辑的电压反相器硬件元件来解决。它惟一的功能是当输入为高时产生低输出，当输入为低时产生高输出，因此，将变量 LT 变为 HT，反之亦然。用来表示该元器件的混合逻辑符号如图 4-8 所示，这里同斜划线标志的情况一样，有 1 个相关的半箭头。

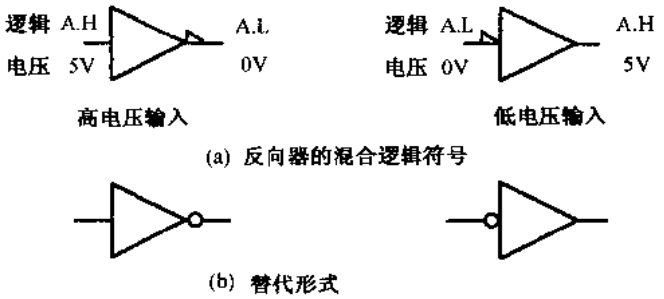


图 4-8

正如前面已明显指出的一样，许多设计者喜欢用小圆圈来代表 LT，如图 4-4 和 4-8(b)所示。因为关于这个约定没有统一的安排，在一些问题上，我们将用小圆圈来指明 LT(而不是逻辑反)，在其他问题上用半箭头。我们已极小心地避免用小圆圈来指明逻辑反，除非已清楚的声明使用正向逻辑规则。

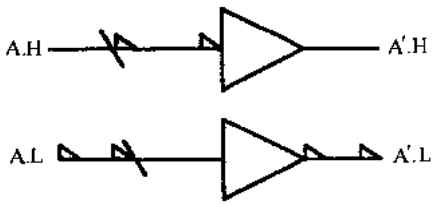


图 4-9

在混合逻辑规则中，我们将硬件和逻辑功能分开，因此将 A.H 变为 A′H 或将 A.L 变为 A′L 的操作图形化由两项来表示：表明逻辑转换的斜划线(逻辑)和表明所需的 LT-HT 转换的三角形(硬件)。正确的混合逻辑符号如图 4-9 所示。

提示：除了与斜划线和反相器相联系的半箭头外，也将其他的半箭头加到线上，与 LT 变量相一致。这导致了它们出现的每一条线段上有两个半箭头。在正确构建的混合逻辑图中，在任何给定的线段上半箭头总是平衡的，这是一个重要原则。

可用图 4-10 所示的与非硬件或者与或硬件来实现电压反相器。所示的每个电路都满足电压反相器的要求，因为它有 HT 输入和 LT 输出，并且没有实现逻辑。可利用布尔等式 A·A = A，A·1 = A，和(A + 0) = A 来推导后者特性。

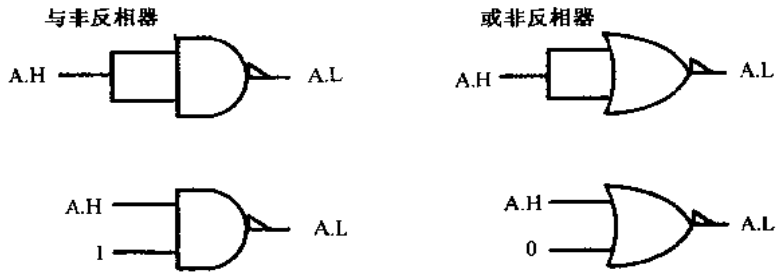


图 4-10

例 4.4 反相器的使用。假设要求实现逻辑函数 F = A′BC + (BCD)′，其中变量 A，B，C 和 D 都以 HT 的形式出现，并且 F 也为 HT。这就要用前面 4.3 节描述过的与非硬件。

第 1 步：在正确的地方用斜划线构建逻辑图（见图 4-11）。

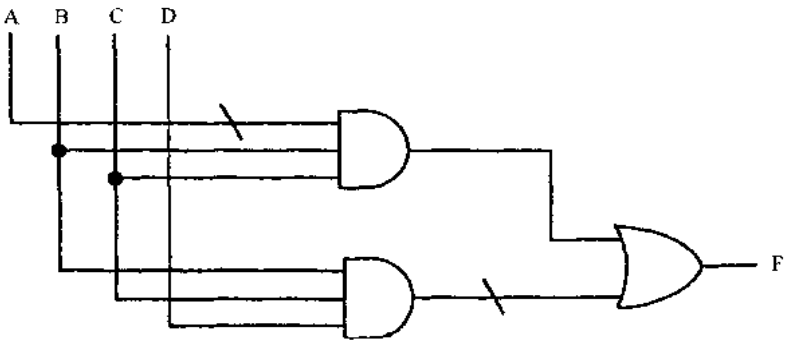


图 4-11

第 2 步：在输入和/或输出端插入适合于所选硬件的半箭头。记住肯定存在与每一条斜线相关的半箭头。（见图 4-12）

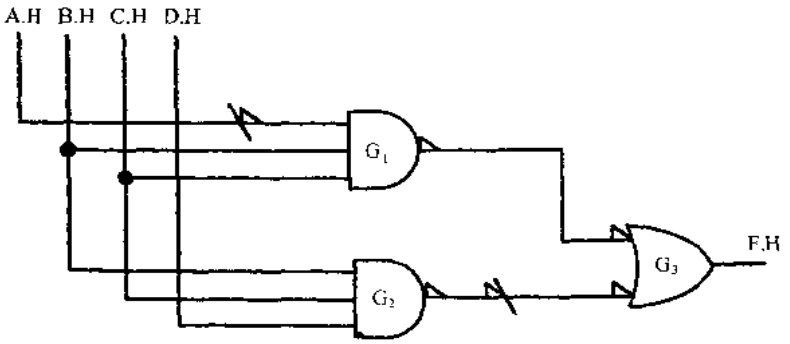


图 4-12

第 3 步：在任何连接的线段上注意半箭头的不平衡性，从而确定需要电压反相器。本例中有两个这样的情况：在 G_1 的输入线上有 A′反相和界于 G_3 的输入和斜划线之间的线段。这些不平衡通过添加电压反相器 G_4 和 G_5 来解决，如图 4-13 所示。提示：考虑最后的结果时，应该将与斜划线相关的半箭头放在两侧；这两种情况下对于硬件反相器都有这种需求。总之，保持半箭头与 LT 变量关系是很好的做法。

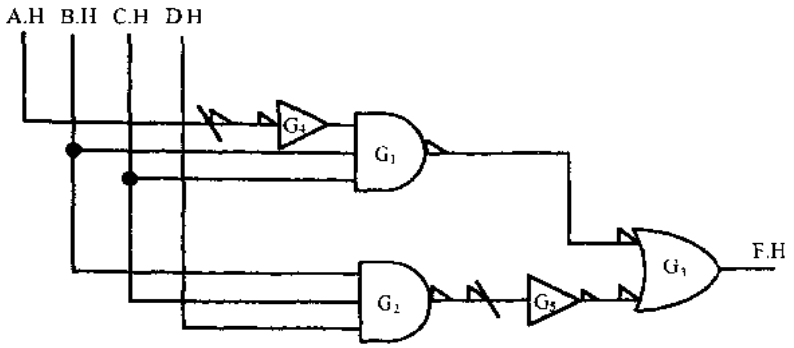


图 4-13

第 4 步：将混合逻辑图转化为线路图。去掉斜划线，因为它们仅代表逻辑操作而不是硬件元器件。如附录 B 所示，将双列直插封装的管脚连到硬件逻辑器件的输入和输出上。将 G_1，

G_2, G_3 门选为 3 输入与非门芯片 74HC10 的元器件，而 G_4 和 G_5 是 74HC04 片上 6 个反相器中的两个。当把管脚号添加到图中时，电路就易于连线了（见图 4-14）。提示：G_3 是 3 输入与非门，通过将管脚 9 和 10 连在一起就将其转化成了 2 输入与非门。

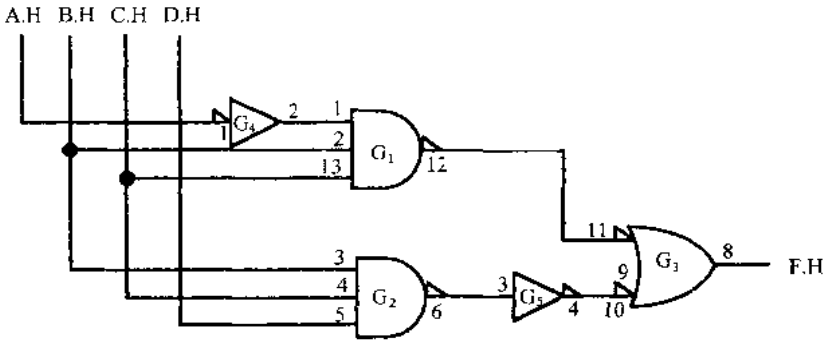


图 4-14

例 4.5　用于非硬件实现函数 $F = E + A'B(C + D)'$，变量 A, B, D 和 E 为 HT，而 C 和 F 为 LT。

第 1 步：按斜划线规则画出逻辑图。（见图 4-15）

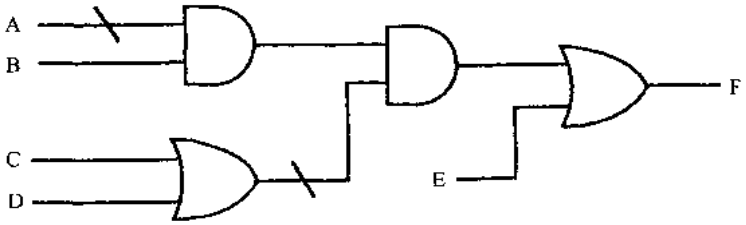


图 4-15

第 2 步：添加与逻辑反相连的合适的半箭头，重复地标出所选硬件的任何 LT 输入和输出（见图 4-16）。

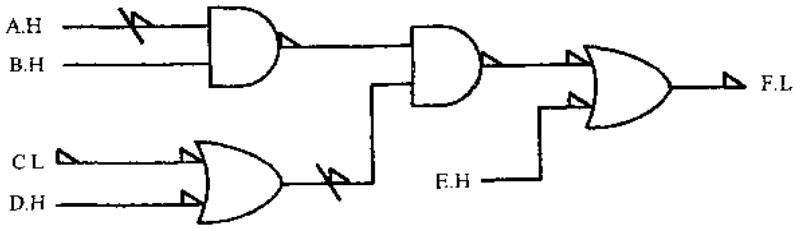


图 4-16　添加了半箭头的逻辑图

第 3 步：用电压反相器平衡半箭头（见图 4-17）。

第 4 步：转化成线路图（见图 4-18）。

从 G_1 到 G_4 取自于同一片四-2 输入与非门芯片 7400；从 G_5 到 G_{10} 取自于一片 16 进制转换器 7404。

提示：G_6, G_7, G_9 和 G_{10} 说明了电压反相器的使用，这里没有相应的逻辑取反，这是重要的一点。

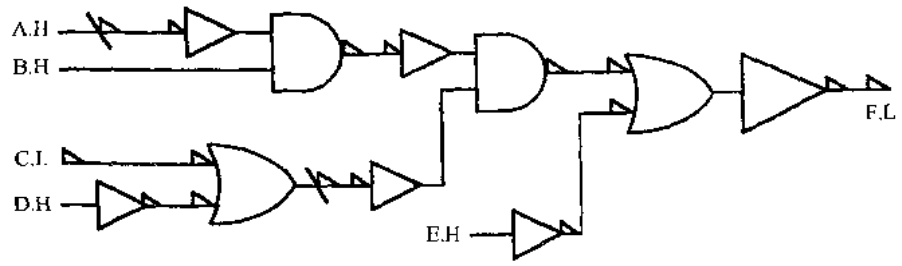


图 4-17　添加了电压反相器的逻辑图

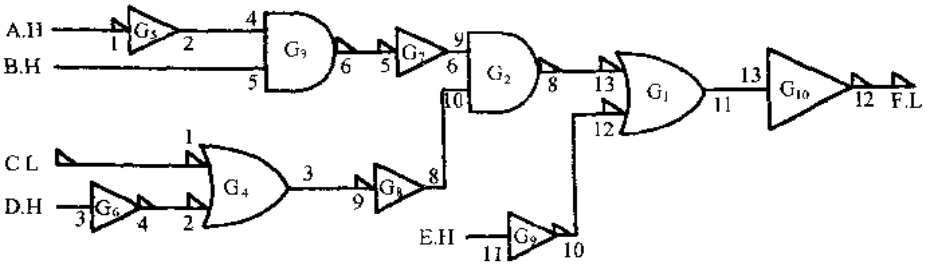


图 4-18　去掉了斜划线的连线图

利用异或电路的实现

当使用硬件元器件而不是标准门电路时，逻辑设计者必须从硬件矢量中确定与输入输出相连的真值。设计者按要求画出半箭头并且如上处理。通用异或电路硬件通常具有 HT 输入和输出，而异或非电路极有可能有 HT 输入和 LT 输出。

4.4　混合逻辑的描述规则

混合逻辑规则的优点之一是直接来自于线路图的电路，它可以很容易地确定设计者要完成一个什么样的逻辑，这是非常有用的！为了从混合逻辑图中发现内在的布尔表达式，仅需去掉所有的半箭头和反相器，然后直接从图中读取逻辑。通过逆向观察图 4-15 到 4-17 可以很容易地研究出这个过程。

虽然设计者在逻辑硬件的说明中经常利用混合逻辑的概念，但并不是所有设计者都使用 4.3 节中说明的技术。在这样情况下，仅仅将存在于电路中的硬件在图中示出，而删去逻辑反作为分开的符号。

例 4.6　考虑例 4.4 去掉斜划线的线路图(图 4-14)。通过假定逻辑取反，在任何带有不平衡半箭头的线段上可以确定出内在的布尔逻辑。因此，参考图 4-19，发现逻辑取反发生在 A 和 G_4 间的线段及 G_2 和 G_3 间的线段上，然而，在 G_1 和 G_3 间的线段及 G_5 和 G_3 间的线段没有发生逻辑取反。要记住的一点就是在混合逻辑规则中，可将一组不平衡的半箭头认为是逻辑取反。

例 4.7　正向逻辑规则仅仅描述硬件，假设所有信号是 HT 并且用小圆圈来表明电压取反。在这个常见的方案中，将与非硬件用单独的符号描述，如图 4-1 所示，而不管它的逻辑目的。将这种可替代的规则用在图 4-19 的电路上，我们得到图 4-20 所示的结果。通过使用摩根定律能够得到函数 F 的期望表达式。

提示：在正逻辑规则中，所有的电压反相器完成逻辑取反。因为布尔运算可得到或函数，

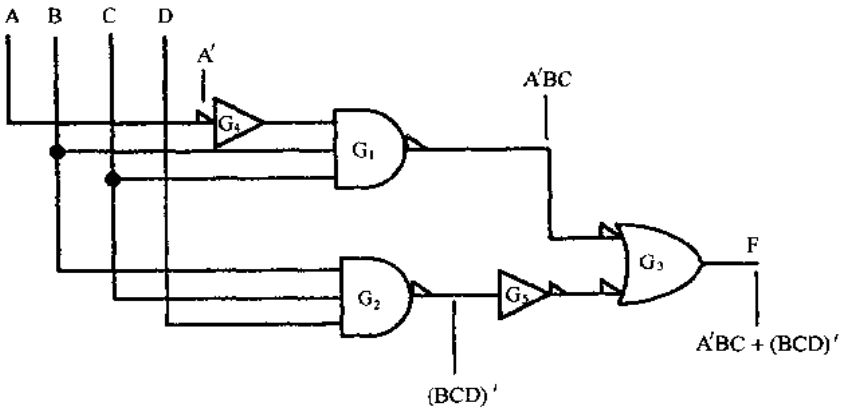


图 4-19

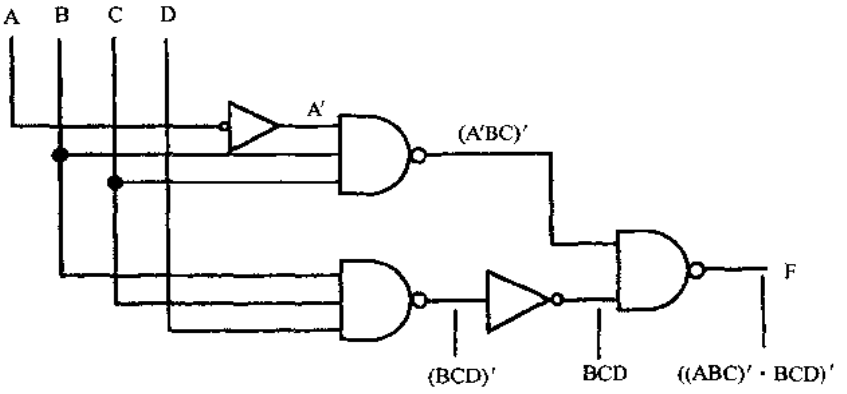


图 4-20

所以应该清楚的看到例 4.7 所示的正逻辑本身不能很好地适合已确认的内在逻辑。

注意：目前使用的几种逻辑规则是很重要的，并且学生应该非常小心地理解任何资料中使用的特定的一种。花点时间用在避免错误上是好的。

4.5　混合逻辑在解决冲突时的使用

混合逻辑规则除了用来确定物理逻辑实现中电压反相器的位置，也允许在设计过程中保存逻辑图。半箭头非常有用，因为可用它来确定电路特定点上与真相联系的电压。这是非常有用的信息，因为冲突通常发生在电压上而不是逻辑上。

例 4.8

(a)在图 4-21 中，输入逻辑信号 W，X 和 Y 如图所示随着时间变化，画出 A，B 点和作为时间函数的 F 的逻辑输出。

去掉半箭头和反相器(不完成任何逻辑)，可看出 A = WX，B = WX + Y，和 F = Y(WX + Y)。布尔代数告诉我们可将函数 F 简化为 F = W，聪明的学生会认识到在硬件上用一根电线就可以实现它！和作为参考的输入一起，所求的波形如图 4-22 所示。

(b)如果使用的逻辑分析仪将所有信号看作高有效信号(HT)，画出在输入端和在 A、B、F 点观察到的电压波形。

因为 Y，A 和 F 是 LT，这些点处的电压波形相对于它们的逻辑将被取反(见图 4-23)。

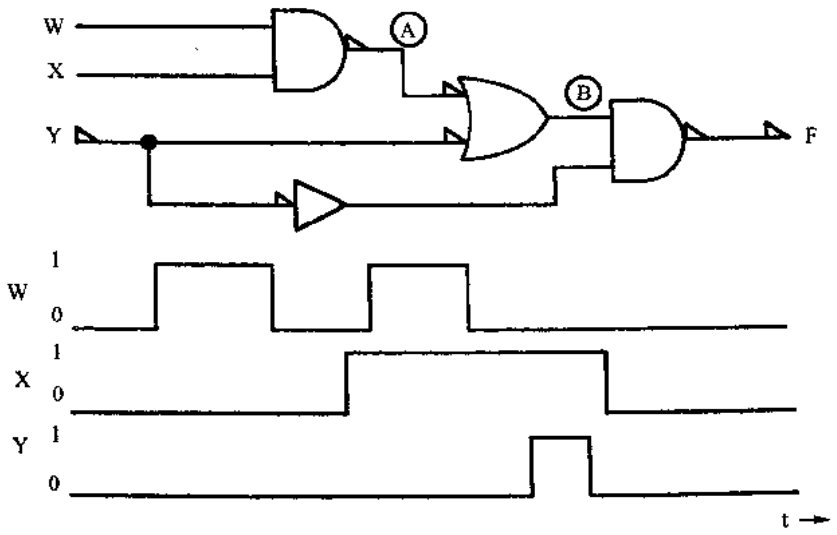


图 4-21

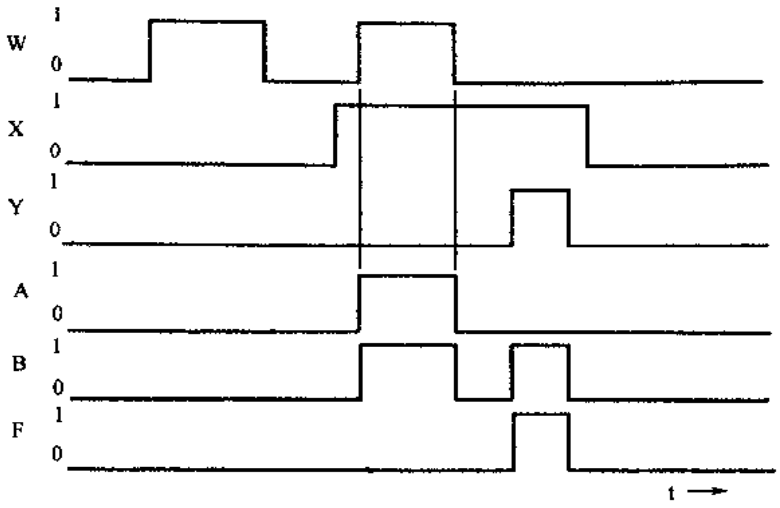


图 4-22

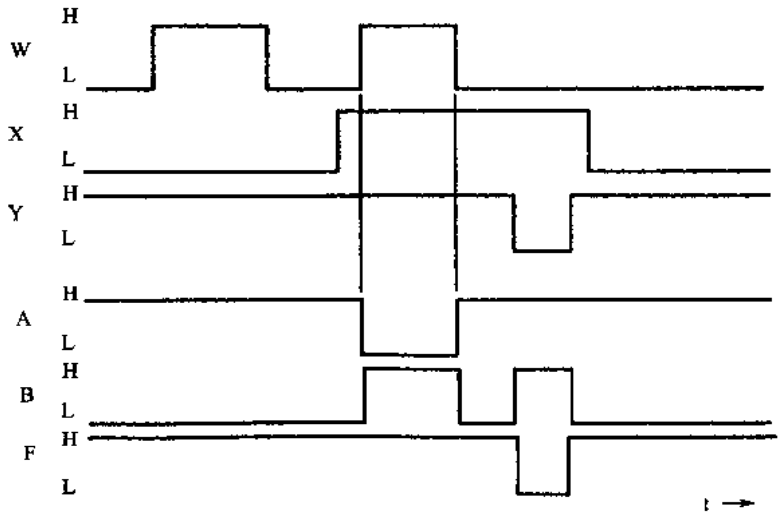


图 4-23

习题解答

4.1　或非硬件定义为当且仅当所有输入为低时输出为高。画出 3 输入情况下的电压 I/O 表。并且，为用作与门和用作或门的器件创建合适的真值表和符号(使用半箭头规则)。

解　或非 I/O 真值表如图 4-24 所示。画出 LT 输入和 1 个 HT 输出，产生了满足逻辑与定义如图 4-25 所示的逻辑真值表。画出 HT 输入和 LT 输出，产生相应于或的真值表(见图 4-26)。

输入			输出
A	B	C	F
L	L	L	H
L	L	H	L
L	H	L	L
L	H	H	L
H	L	L	L
H	L	H	L
H	H	L	L
H	H	H	L



或非硬件符号

图 4-24

输入			输出
A	B	C	F
1	1	1	1
1	1	0	0
1	0	1	0
1	1	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0



用作与门的或非硬件

图 4-25

输入			输出
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



用作或门的或非硬件

图 4-26

4.2　一数字硬件电路的特性由表 4.3 所示的逻辑电平表定义。指出如何分配真值使电路作为逻辑与，逻辑或和电压反相器。

表 4.3

输入		输出
A	B	G
L	L	L
L	H	L
H	L	H
H	H	L

解　对于与函数，仅有一种情况输出为真，因此必须选第 3 行。函数 G 很明显是 HT，接着必须指定输入 B 为 LT，输入 A 为 HT，创建表 4.4 所示的逻辑与真值表。参考表 4.3，我们知道对于或函数，G 和 A 必须是 LT，而 B 是 HT。最后，如果保持 A 为高，那么 G 将是 B 相反的电压值，满足反相器的要求(见表 4.5)。

表 4.4

输入		输出
A	B	G
F	T	F
F	F	F
T	T	T
T	F	F

表 4.5

输入		输出
A	B	G
H	L	H
H	H	L

4.3　将 3 输入或非门封装在一块芯片上。尽量有效地组合这些门来创建一个相当于 5 输入的或门。使用输入输出最方便的真值。

解　两个 3 输入或门可以组合成一个相应的 5 输入或门(图 4-27)。或非门由图 4-28 所示实现。要求与门之间用半箭头来平衡的反相器可以试着通过将一个或非门的输入连在一起来得到，如图 4-29所示。这就需要一个芯片：输入是 HT，输出是 LT。

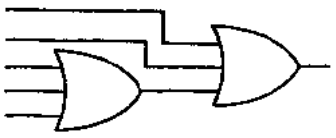


图 4-27

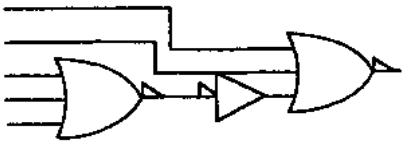


图 4-28

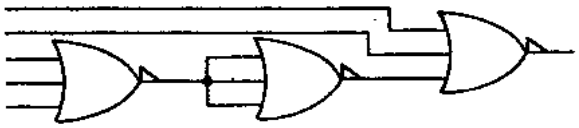


图 4-29

4.4　用或非硬件实现例 4.4 的逻辑函数。

解　一个纯粹的逻辑图是与硬件无关的。然而专用硬件混合逻辑图，加入半箭头可以标明合适的真值，如图 4-30 所示。为了实现其功能，我们平衡箭头从而确定任何需要电压反相器的位置。提示：或非的实现需要比例 4.4 的与非实现多 3 个反相器。

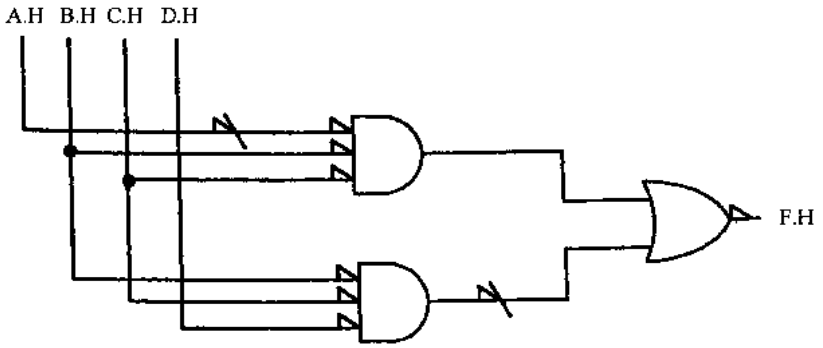


图 4-30　创建或非混合逻辑图的第一步

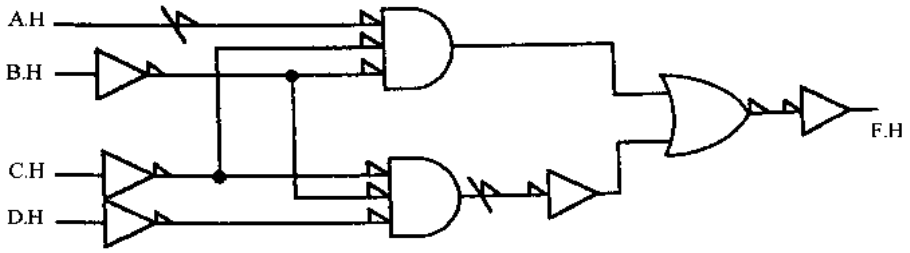


图 4-31　最终的或非混合—逻辑图

4.5　利用与非硬件实现第 2.13 题的全加器(见图 4-32)。

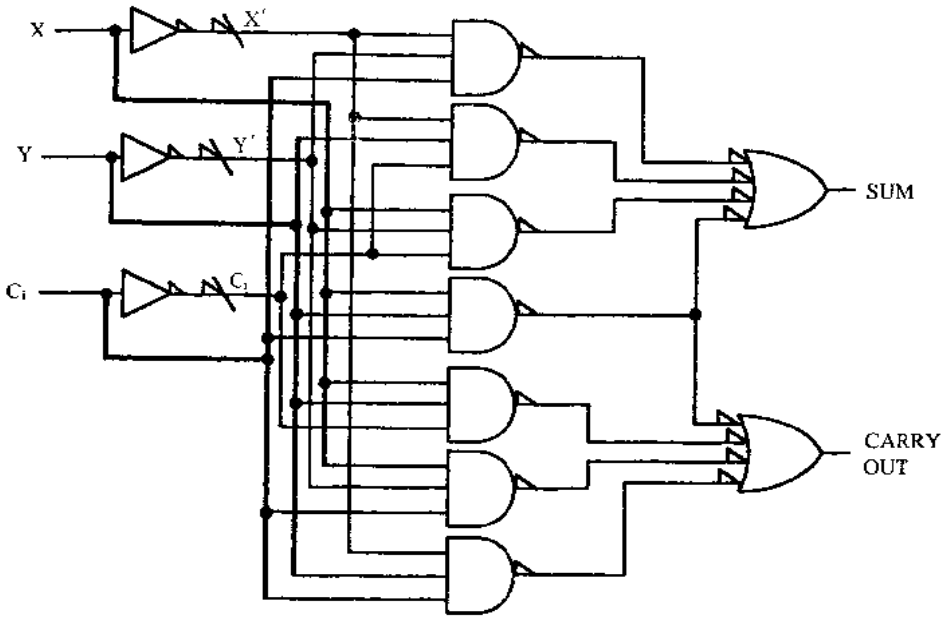


图 4-32

4.6　利用与非硬件实现第 2.15 题的全加器，并且在适当的地方用异或硬件。假设 A, B, C_i, S 和 C_o 都是 HT。

解　参考图 2-13，按需要添加半箭头(见图 4-33)。因为所有半箭头都平衡了，所以不需要反相器。

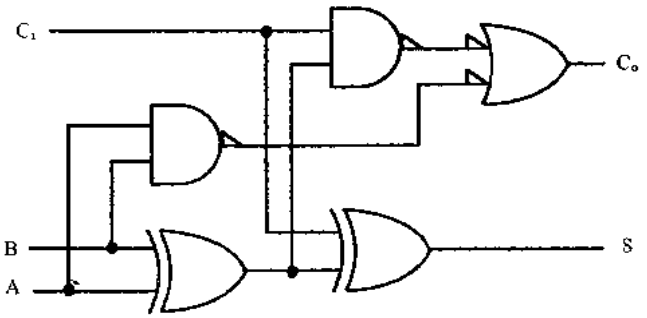


图 4-33

4.7　用异或非代替异或重复第 4.6 题。记住异或非具有 LT 输出。

解　见图 4-34，提示：需要添加两个反相器。

4.8　对与门和或门利用或非硬件来重复第 4.6 题。

解　见图 4-35，提示：在这种情况下，需要 5 个反相器。

4.9　A, B 和 C_o 都是 LT 而 C_i 和 S 是 HT，重复第 4.6 题。

4.10　假设正逻辑硬件图如图 4-37 所示，画出 3 个不同的混合逻辑等效电路图。

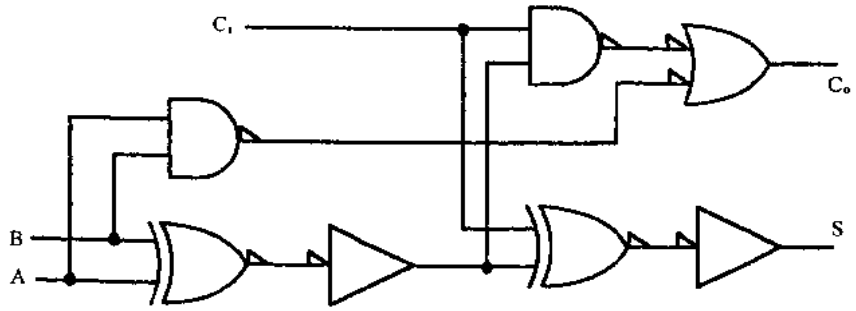


图 4-34

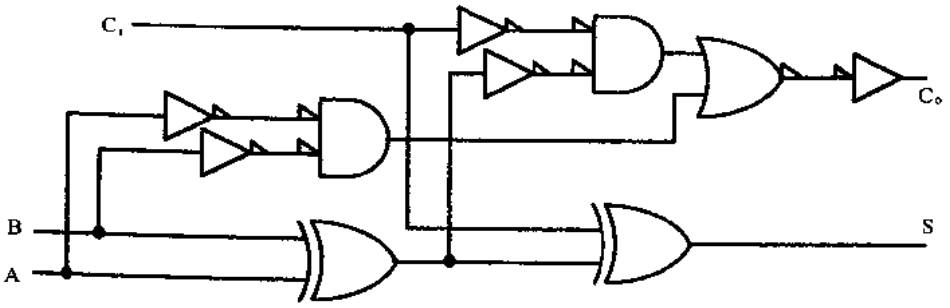


图 4-35

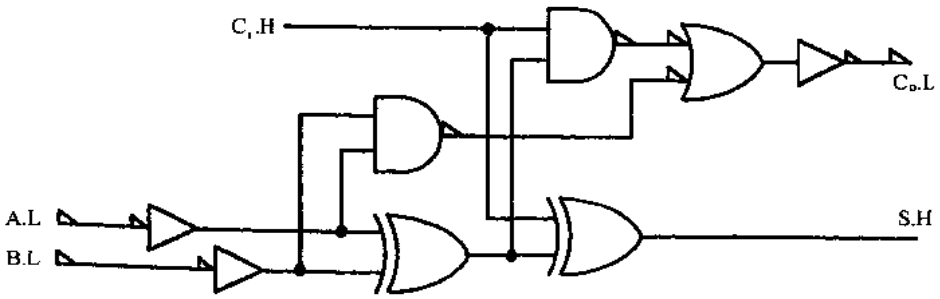


图 4-36

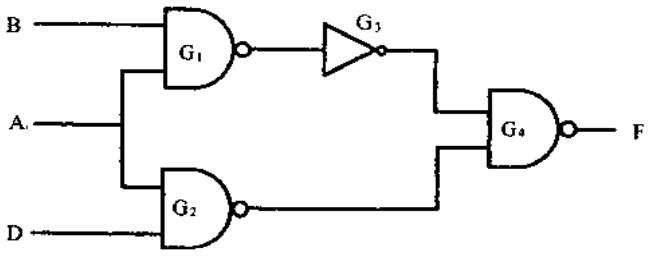


图 4-37

解 如果需要，能够借助于摩根定律确定内在的逻辑，如图 4-38 所示。虽然这不是必要的步骤，出于比较的目的和强调如何将输出表达为多于一种方式是比较方便的。记住在混合逻辑表示中，所有硬件反相器可以被组合的斜划线标志和 LT 到 HT 的转换器替代：

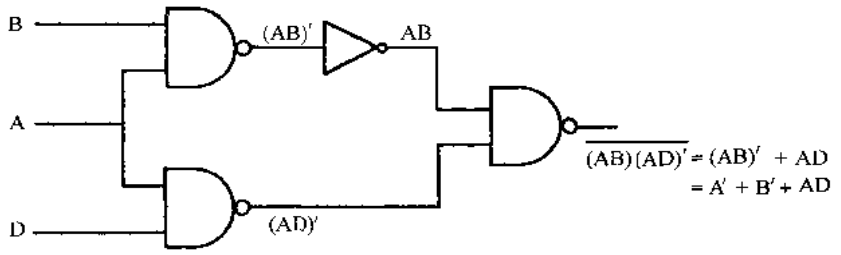
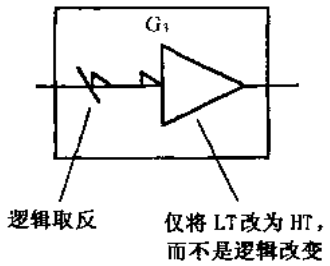


图 4-38



现在已经有了几种选择，下面讨论其中 3 种。

1．假设所有与门正在实现或功能。这引出了图 4-39 所示的安排。提示：与 G_3 相关连的斜划线在 G_1 和 G_4 之间添加了一个半箭头，而另一方面，在它的正下方的支路中缺少一个半箭头。为改正这些不平衡，移动斜划线，在输入端再加上斜划线，调整半箭头的位置，如图 4-40 所示。这个情况很清楚地说明了反相过程的两个解释是如何实现的。反相器 G_3 仅将 HT 转化为 LT，而没有逻辑的改变。而它下面的惟一的斜划线表明了发生在 G_2 和 G_4 之间的逻辑取反过程。这里，在思维中我们将 G_2 .H 转化成相应的 G_2' .L，实际上，这两个是相同电平的不同解释。

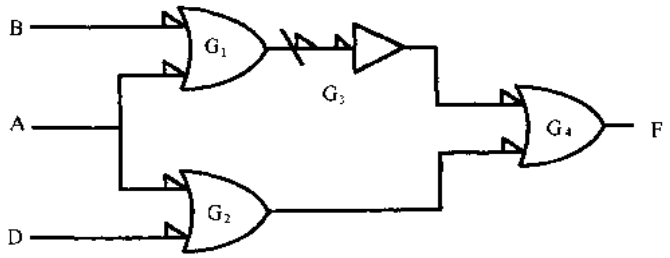


图 4-39

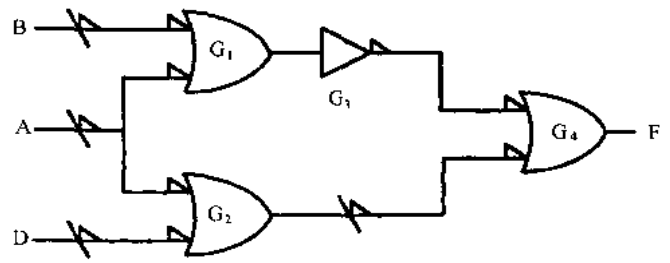


图 4-40

去掉半箭头和反相器发现了内在的逻辑图(图 4-41)，从中我们可以读到

$$F = (A' + B') + \overline{A' + D'} = A' + B' + AD$$

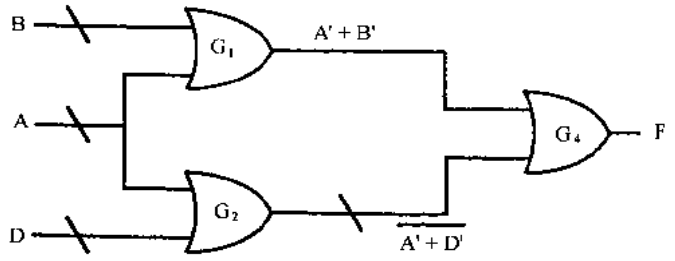


图 4-41

2. 假设所有与非门正在实现与功能。这引出了图 4-42 所示的安排。现在我们根据需要添加或去掉斜划线标志来平衡半箭头。提示:这些组件仅仅是逻辑上的,不会改变基本硬件配置。参考图 4-43。

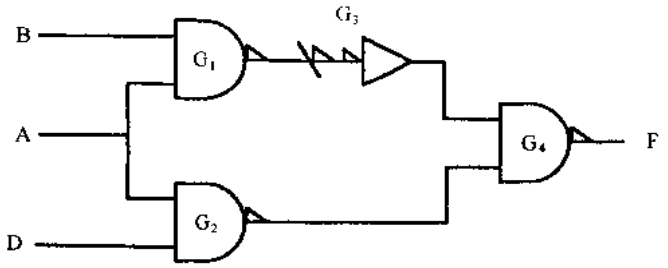


图 4-42

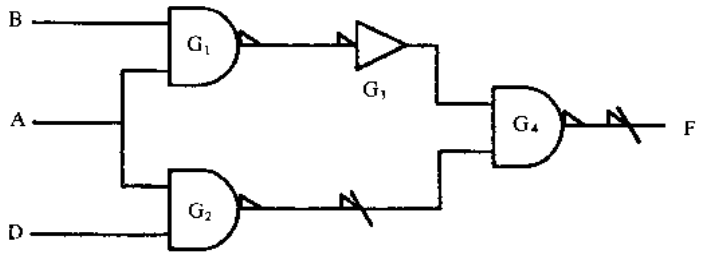


图 4-43

去掉反相器和所有半箭头之后,可看出输出的布尔表达式是 $F = \overline{(AB)\cdot(AD)'} = (AB)' + AD$,它和前面的结果是一样的。

3. 假定 G_1 和 G_2 执行与功能,G_4 是或功能,这一点已从图 4-38 的结果推导出。如我们在图4-44中所看到的,所有半箭头都平衡了,并且不需要进一步的运算。这是最简化的形式而且通过观察得到了所希望的函数。当需要预测波形和研究电路延迟的效果时(见第 6 章),使正向逻辑硬件和混合逻辑间能转化是非常有用的。

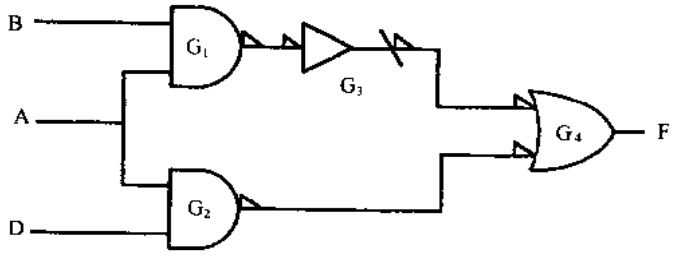


图 4-44

4.11 随着电路复杂性的提高,从正逻辑硬件图中解释出内在的逻辑相对难度明显增加。图

4-45 给出了例 4.5，图 4-17 中的电路的正逻辑的形式。确定所实现的逻辑。

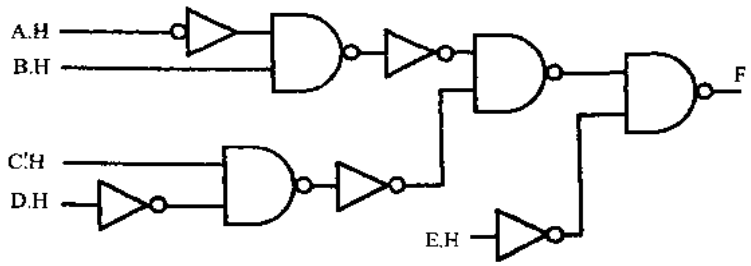


图 4-45

解 ☞ 提示：输入变量 C.L 已被与它相等的 C′.H 替代，并且已经指明了 HT 输出，因为在正向逻辑约定中只允许 HT 变量。在图中写出中间表达式，得到了图 4-46 所示的结果，可用它来得到如下要求的函数：

$$F=\overline{E'(A'B+C'D')}=E+\overline{A'B+C'D'}=E+A'B\cdot C'D'=E+A'B(C+D)'$$

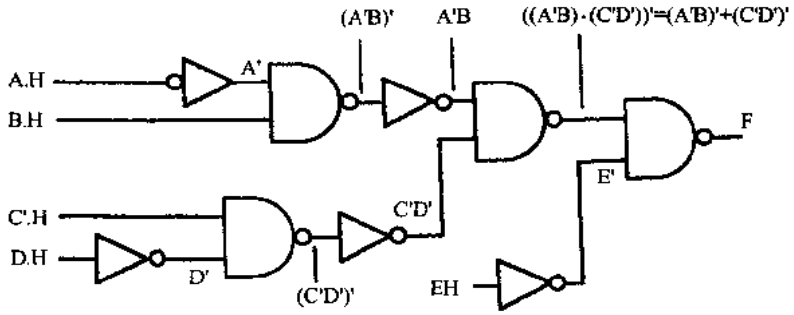


图 4-46

将解出图 4-15 的逻辑和解本练习的逻辑相比较是有指导性意义的。

4.12 图 4-47 的逻辑具有图 4-48 所示的随时间变化的输入。假设电路同时响应输入变化，画出输出电压波形图。变量 A, B, C 和 D 都是 HT。

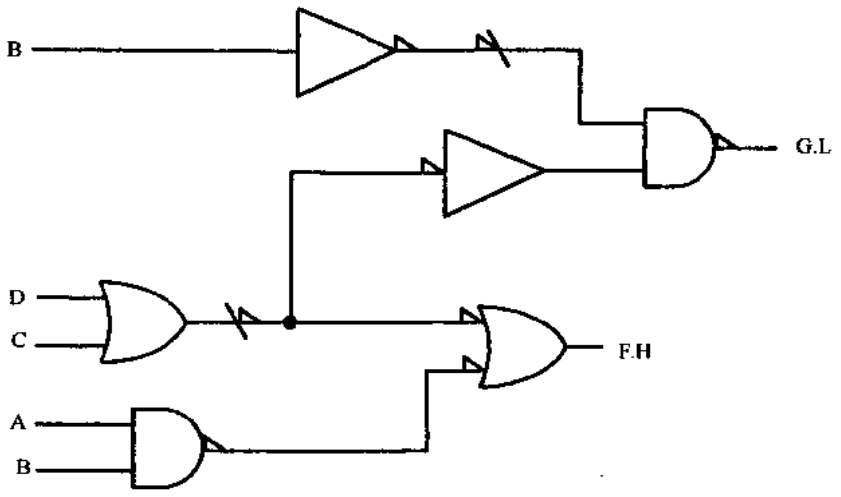


图 4-47

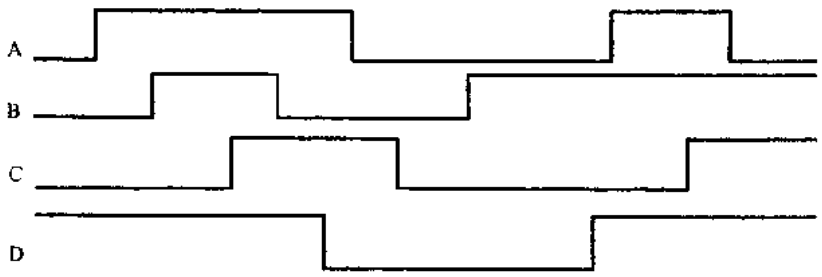


图 4-48

解 从逻辑图中移去反相器和半箭头之后，能够很容易地确定 F = AB + (C + D)′，对其第 2 项利用摩根定律后，产生了 F = AB + C′D′。也就是，当 A 和 B 都为真或 C 和 D 都为假时 F 为真。而且，从逻辑图中可看出，G = B′(C + D)′，或者，与其相等的有 G = B′C′D′。因此，只有当 B, C 和 D 都为假时 G 为真。

通过设置 1 为真，0 为假得到所要求的输出逻辑波形，并且在点对点的基础上利用布尔方程式，如图 4-49 所示。因为将 G 指定为 LT，所以一定要将 G 的波形取反来获得期望的电压输出。

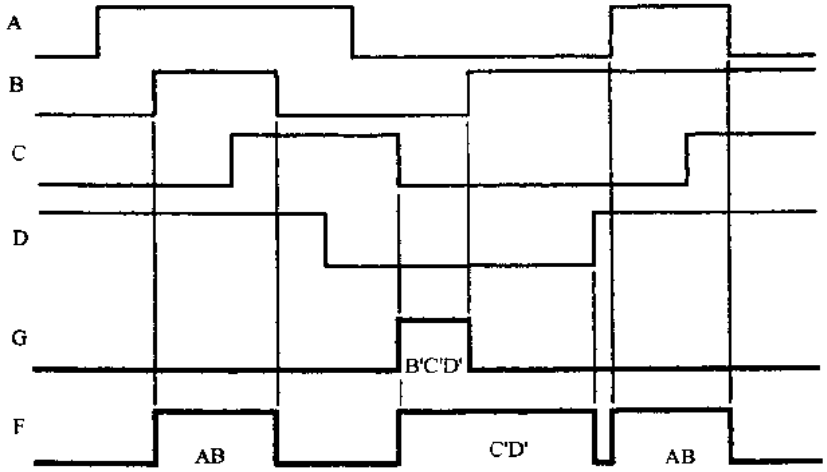


图 4-49

4.13 设计具有 3 输入端 A, B, C 的电路，无论何时，任何两个或多于两个输入为真时输出 (F)为真。假定 A 和 C 是 HT，而 B 和 F 是 LT。见图 4-50。

(a)画出卡诺图，选择最好的包围 1 的方法。

(b)利用或非硬件实现器件的设计。

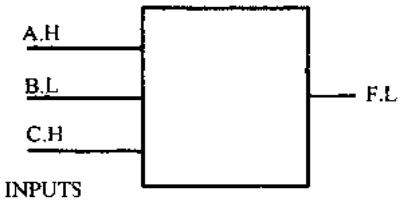


图 4-50

解 (a)对图中的每一个方格的坐标求 1 的和，并且任何方格坐标值加到 2 或多于 2 时在方格

中置 1。

(b)添加半箭头和必要的电压反相器，得到图 4-52。

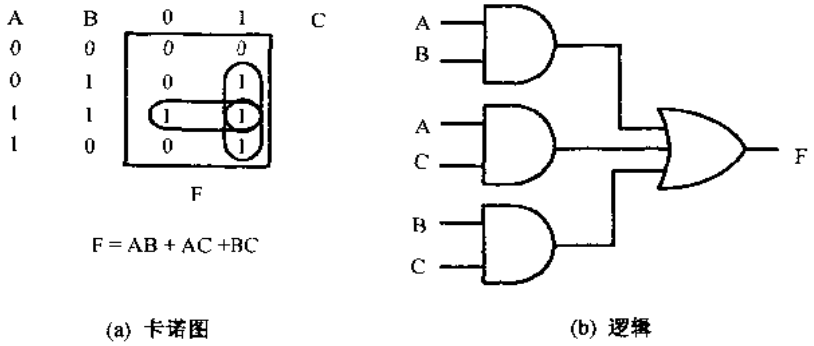


图 4-51

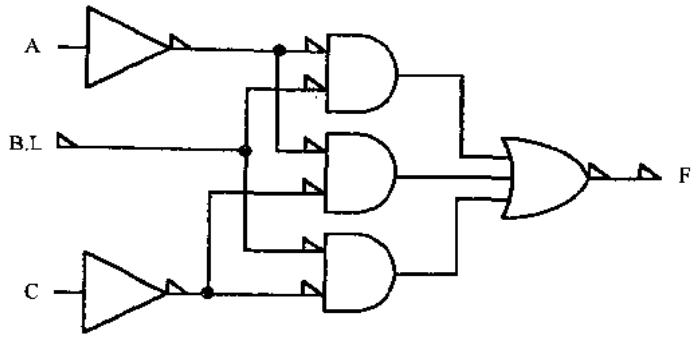


图 4-52

4.14　按要求重复题 4.13：

(a)包围 1 的与非实现。

(b)包围 0 的与非实现。

解　(a) 逻辑图与图 4.51 所示的逻辑图一样；它的与非实现如图 4-53 所示。

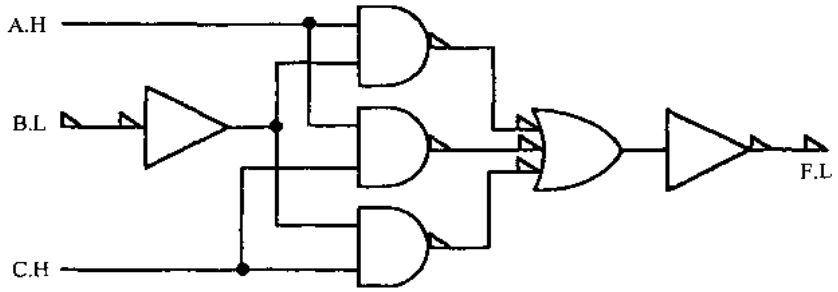


图 4-53

(b) 从包围 0 中可得到的最简表达式是 $F' = A'B' + A'C' + B'C'$。相应的逻辑图如图 4-54 所示，与非实现如图 4-55 所示。注意，通过比较图 4-52，4-53 和 4-54，可看出硬件和包围方式的选择如何影响反相器的数量和位置。

4.15　设计一个逻辑来检验两个二进制数(A_2A_1 和 B_2B_1)，当数字 B_2B_1 比 A_2A_1 大或相等时，产生真的输出。变量 A_1，A_2 都是 LT 而 B_2，B_1 和输出都是 HT。用与非硬件实现。

解　按指定的要求直接画出卡诺图，并且用正确的包围 1 的方法，如图 4-56 和 4-57 所示，得到要求的函数。相应的混合逻辑图如图 4-58 所示。

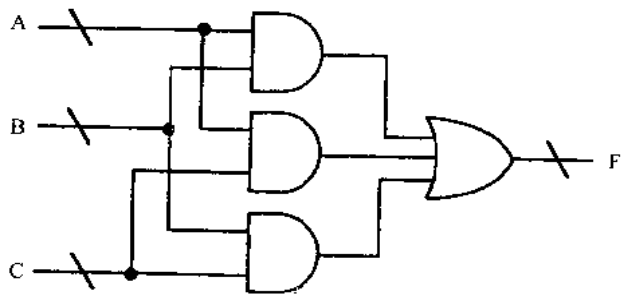


图 4-54

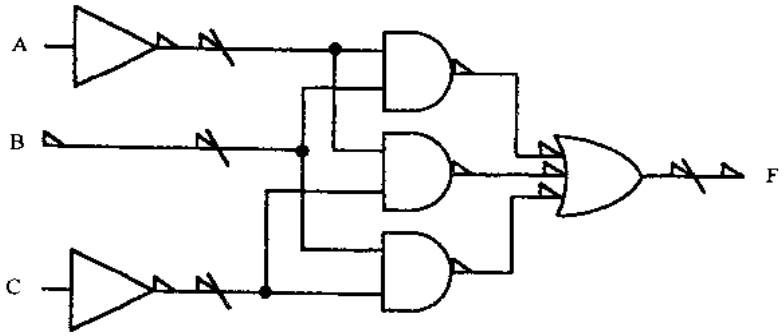


图 4-55

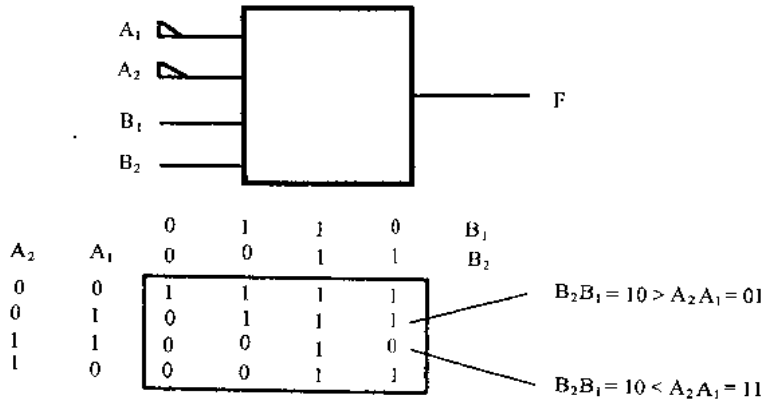
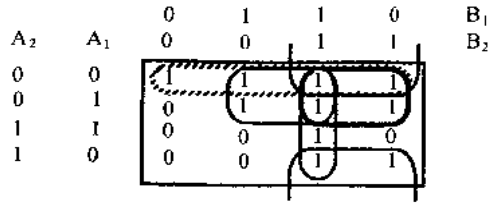


图 4-56



$F = A_2'A_1' + B_2B_1 + A_2'B_1 + A_1'B_2 + A_2'B_2$

图 4-57

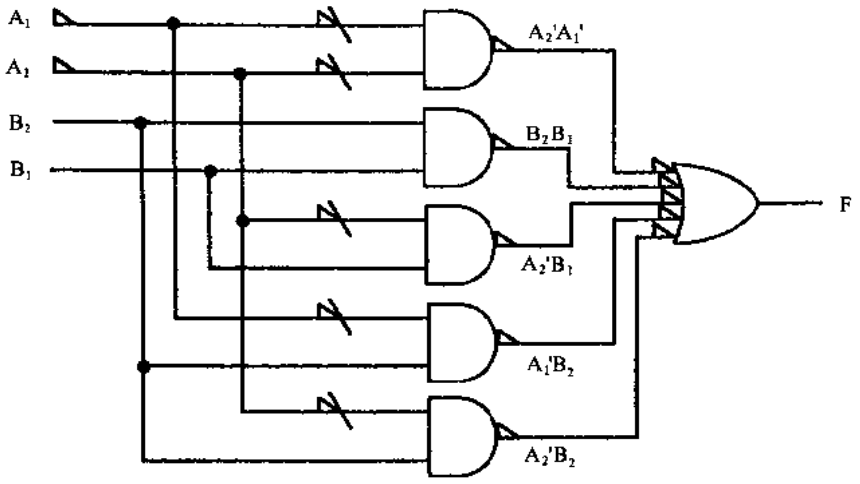


图 4-58

4.16　假设布尔函数 F(A, B, C, D) = [D(A + B′C)(AB + C′)]′，其中 A, B 和 D 都是 HT，而 C 和 F 是 LT，用物理器件完成逻辑。它具有的电压 I/O 表描述如下：当且仅当所有输入为高时输出是高。

解　指定的物理器件特性可以用表 4.6 中的表格形式表示。对输入和输出映射 H = T，得到了如图 4-59 所示的与函数。通过将输入输出映射为 L = T 得到或函数(图 4-60)。逻辑图如图 4-61 所示，并且它的硬件实现使用混合逻辑规则在图 4-62 中给出。注意反相器的增加。根据硬件的选择，每次将与门连到或门上时就需要一个电压反相器，反之亦然。

表 4.6

输入		输出
X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

X	Y	Z
F	F	F
F	T	F
T	F	F
T	T	T

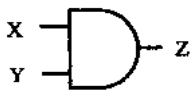


图 4-59

X	Y	Z
T	T	T
T	F	T
F	T	T
F	F	F



图 4-60

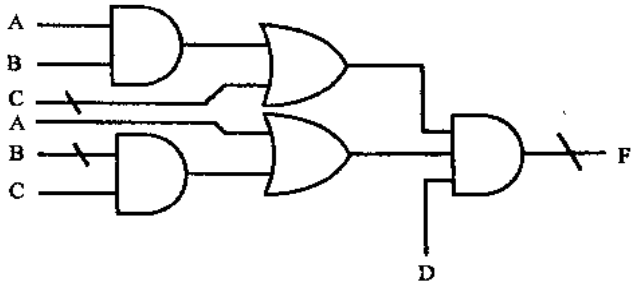


图 4-61

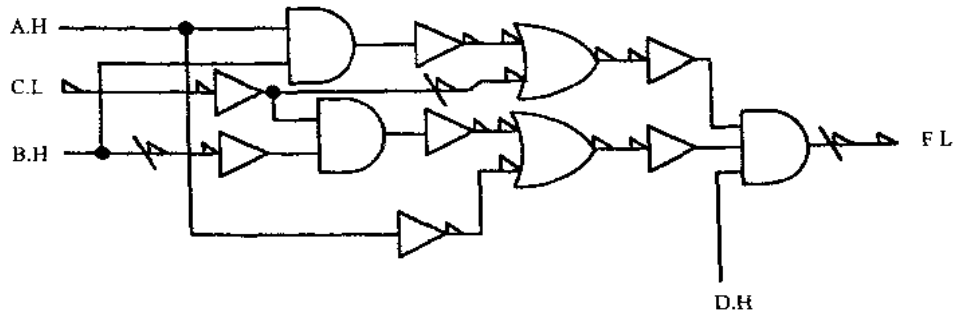


图 4-62

4.17　假设卡诺图如图 4-63 所示。首先用与非硬件和反相器，然后是异或门来实现函数。假定输出和所有的输入都为 HT。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	0	1	0	
0	1	1	0	1	0	
1	1	X	X	0	1	
1	0	X	1	0	1	

F

图 4-63

解　(a)与非实现：利用所有任意项包围 1 的方法产生了 F = AC + C′D′ + AD′ + A′CD。与非实现如 4-64 图所示。

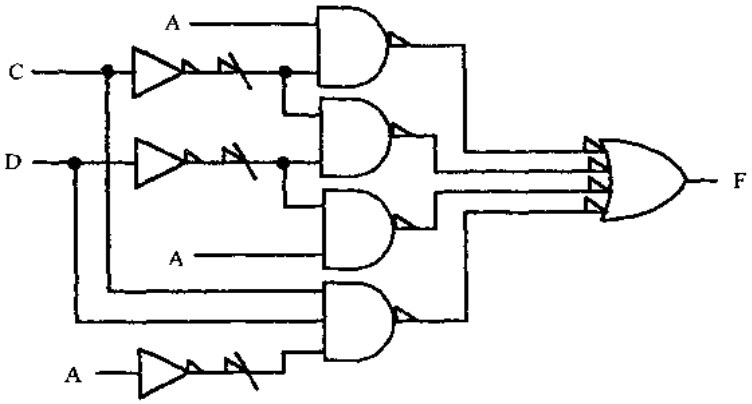


图 4-64

(b)异或实现：通过适当地选择“任意项”，可将卡诺图做成表示异或函数的棋盘形式。也就是第 1 列中的 0 和第 2 列中的 1。结果函数为

$$F = A'C'D' + AC'D + A'CD + ACD'$$
$$= A'(C'D' + CD) + A(C'D + CD')$$

可将括号中的项认为是分别与(C⊕D)′和(C⊕D)相等，如第 3.4 题所示，设 X = (C⊕D)，那么

$$F = A'X' + AX = (A \oplus X)'$$

替代 X 得到了 F = (A⊕C⊕D)′，它可以由图 4-65 的电路实现。提示：同例 2.8 中描述的一样，如何利用第 3 个异或门来得到逻辑取反。

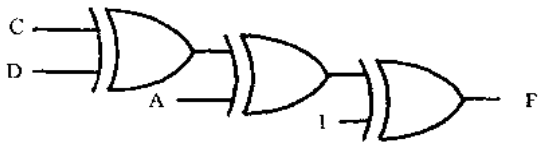


图 4-65

补充习题

4.18 用与非硬件实现函数 F = A′C′D′ + BD′ + BC′。

4.19 假设函数 F 的卡诺图如图 4-66 所示，为了得出最好的包围 1 的方法给任意项分配值，并且画出逻辑图。根据需要，为了减少与非实现所需要的电压反相器的数目，指明输入和输出变量是 LT 还是 HT。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	0	1	X	
0	1	0	0	1	X	
1	1	X	X	X	X	
1	0	1	0	1	X	

F

图 4-66

4.20 用与非硬件实现函数 F = A(B + C′)，在需要的地方用反相器。要求所有变量(输入和输出)都是 HT。

4.21 假设去掉第 4.20 题的与非硬件电路中所有反相器，并且用电线来替代。写出电路产生的正逻辑的布尔表达式，再次假设输入和输出都是 HT。

4.22 假设函数 F = ABC + A′BD′ + BC′D + ABC′，假设真值表中最前面 4 行是无关项，简化逻辑，而且变量以字母升序排列，A 是最高位。画出或非硬件逻辑图。

4.23 实现第 3.22 题所讨论的 7 段译码驱动器的逻辑。

4.24 设计检验两个 2 位二进制数(A_2A_1 和 B_2B_1)的逻辑。当数字 B_2B_1 比 A_2A_1 大时输出为真。用与非硬件实现，并将变量定义为 LT 或 HT 以至于不需要反相器。

4.25 假设卡诺图如图 4-67 所示，用包围 1 的方法简化逻辑，用与非硬件实现。用小圆圈表示 LT 的混合逻辑。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	0	1	1	
0	1	0	1	0	0	
1	1	1	1	X	0	
1	0	:	X	1	1	

图 4-67

4.26 用包围 0 的方法和或非硬件实现第 4.5 题的逻辑。

4.27 用与非硬件实现第 3.25 题的简化的乘法器，确定所需要的小规模集成芯片(SSI)的数目。

4.28 利用 L 的方程式，用与非和异或硬件实现第 2.25 题中的灯控电路。假设开关 A 和 B 为 HT，L 为 LT。比较小规模集成芯片的数目。

4.29 重复第 4.28 题，这一次用 L′的方程式。

4.30 用与非硬件实现第 2.27～2.29 题设计的二进制数减法器，假设输入输出都是 HT。估算芯片数量。

4.31 利用第 3.28 题中讨论的二进制数减法器的简化的布尔方程式，用或非和异或硬件来实现电路，使用 HT 输入输出，将芯片数量与 4.30 题的结果相比较。

4.32 如果输入输出为 LT，重复第 4.31 题。

4.33 确定图 4-68 所示电路表示的逻辑方程式。

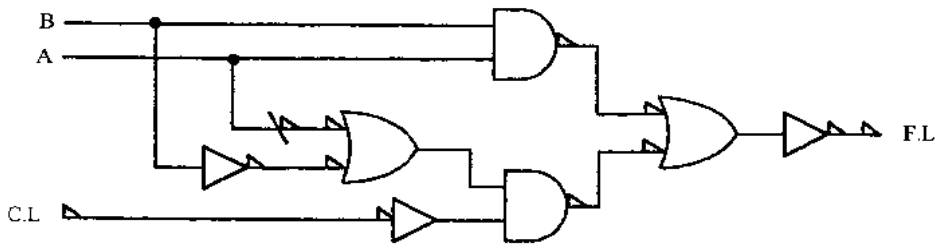


图 4-68

4.34　对第 3.34 和 3.35 题讨论的将 4 位二进制数转化为格雷码的转换器，求出与非表示的输出，估计芯片的数量。

以下 4 个问题讨论了硬件和真值分配如何影响电路的拓扑结构和复杂性。

4.35　用或非硬件实现第 3.38 题的包围 0 的方法。假设输入和输出变量都是 LT。

4.36　用与非硬件实现第 4.35 题的电路，再一次假设输入和输出变量都是 LT，比较结果。

4.37　第 4.36 和第 4.37 题的两个逻辑相同的布尔形式是：

$$F = (B'C + AC + AD)' = A'C' + C'D' + A'B$$

假定 B 和 F 为 LT，而其他变量为 HT，利用与非硬件实现这两种形式。相互比较结果，并比较第 4.36 和第 4.37 题所得到的电路。

4.38　重复第 4.37 题，用或非硬件实现。

4.39　写出图 4-69 所示电路由正逻辑表示的布尔表达式。

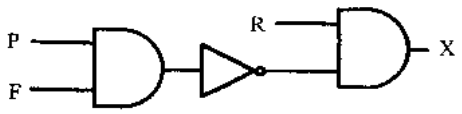


图 4-69

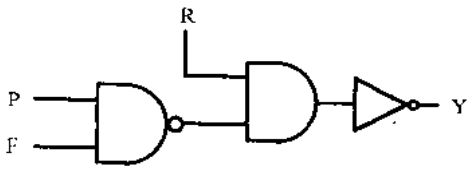


图 4-70

4.40　对图 4-70 所示电路重复第 4.39 题。如果指定 Y 为 LT，指出这个结果和上一个结果的关系。

4.41　画出第 4.40 题中的电路的与非实现的混合逻辑图。

补充习题答案

4.18　见图 4-71。

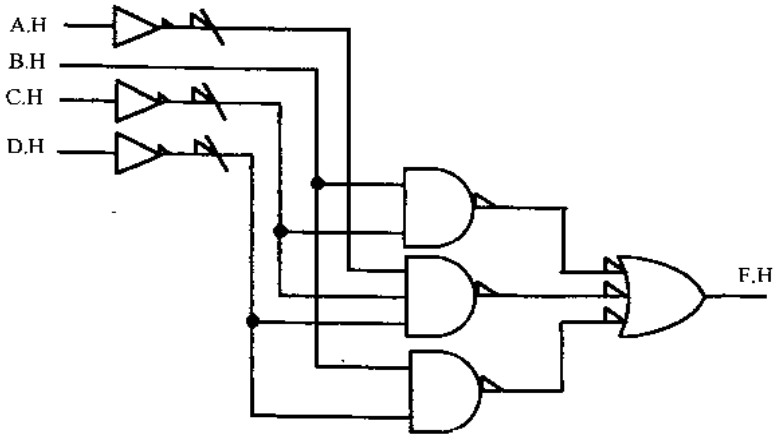


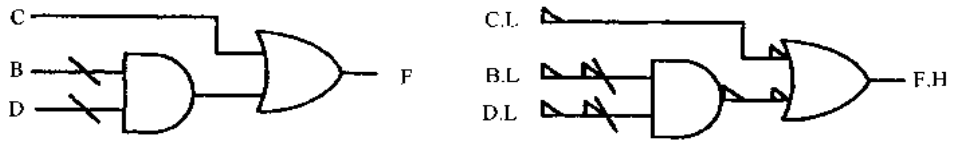
图 4-71

4.19　见图 4-72 和 4-73。图 4-72 所示的圈起来的部分加上 4 个角产生了 F = C + B′D′。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	0	1	X	
0	1	0	0	1	X	
1	1	X	X	X	X	
1	0	1	0	1	X	

F

图 4-72



(a)　逻辑图　　(b) 具有输入为 LT，输出为 HT 并且没有电压

图 4-73　反相器的逻辑实现

4.20　见图 4-74。

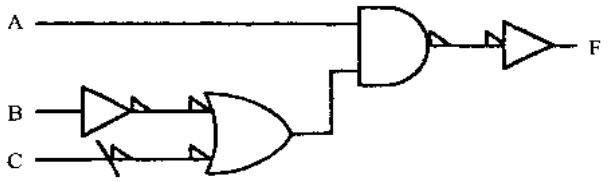


图 4-74

4.21　F = A′ + BC。

4.22　F = AB + BC′ + A′D′和 F = AB + BC′ + BD′都是正确的答案。第 1 种情况的硬件设计如图 4-75 所示。

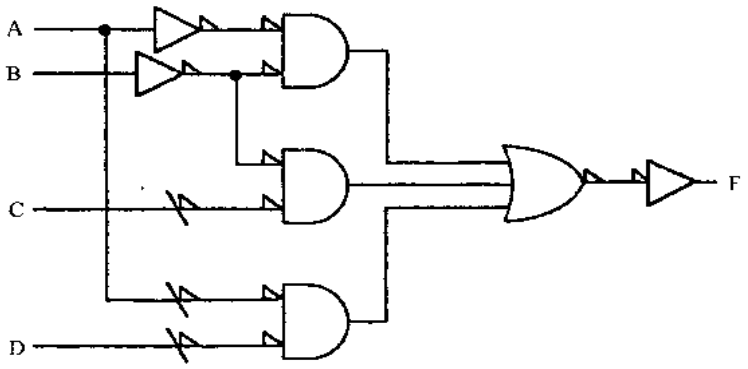


图 4-75

4.23　见图 4-76，提示使用 LT 小圆圈规则。

4.24　按要求直接写出卡诺图得到函数 $F = A_2'B_2 + A_2'A_1'B_1 + A_1'B_2B_1$。所求电路如图 4-77 所示。

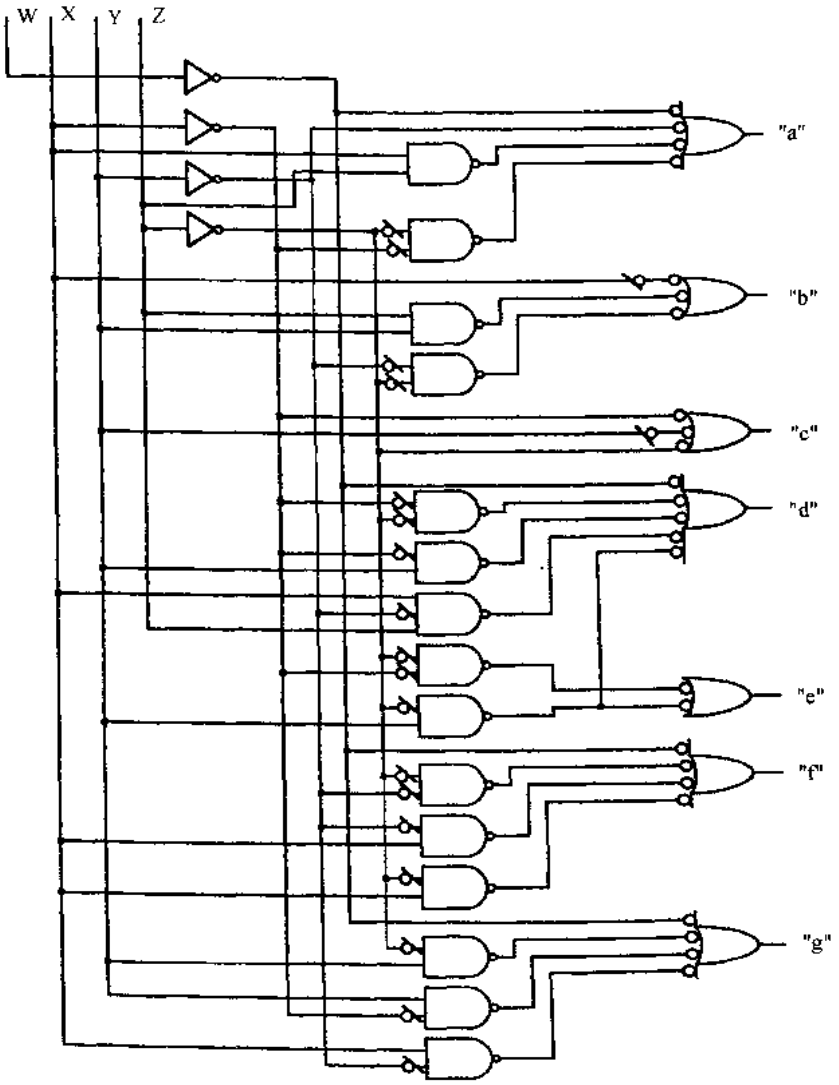


图 4-76

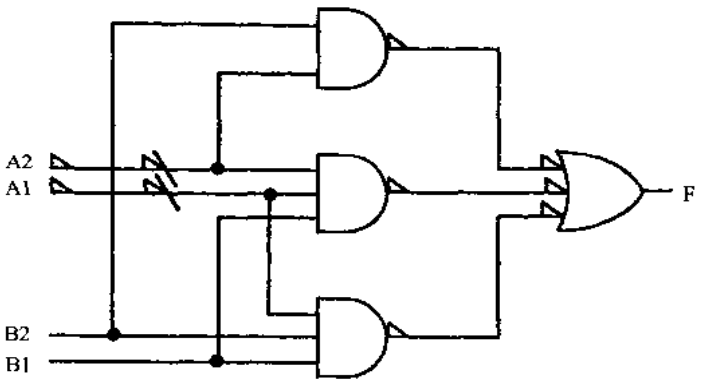
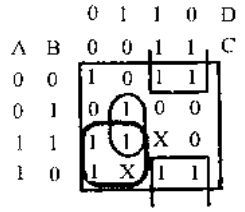


图 4-77

4.25　如图 4-78 所示，化简包含 4 个角的区域，为了清楚起见将它从图中删去。与非实现如图 4-79 所示，其中使用 LT 小圆圈约定。需要使用 3 个芯片(1 个双 4 输入门，1 个四-2 输入门，1 个是 1/3 六倒相器)。



F = B'C + AC' + B'D' + BC'D

图 4-78

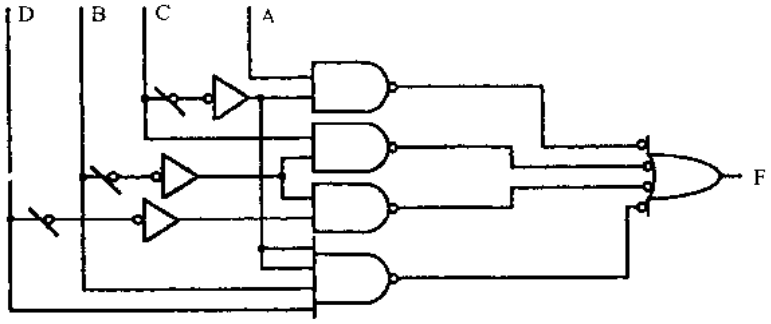
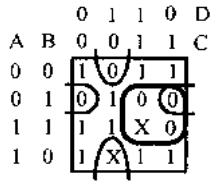


图 4-79

4.26　见图 4-80 和 4-81。需要 2 个芯片(1 个三-3 输入门,1 个四-2 输入门)。



F' = BC + B'C'D + A'BD'

图 4-80

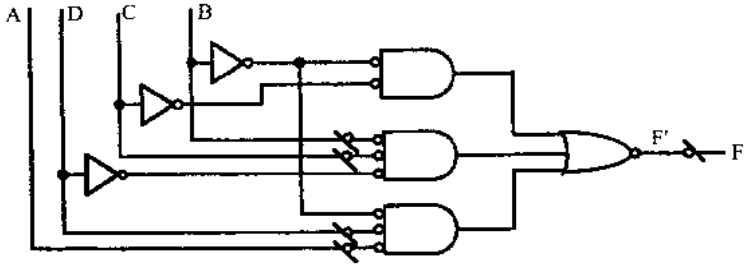


图 4-81　LT 小圆圈规则

4.27　见图 4-82,芯片数目如下:

1 片六角倒相器(包括 6 个门,全用)

1 片四-2 输入门(包括 6 个门,全用)

2 片三-3 输入门(每一个包括 3 个门,全用)

1 片四-2 输入门(包括 4 个门,用 2 个)

总共:5 个芯片

4.28　利用 L = AB + A'B' = (A'B + AB')' = (A⊕B)'得到了图 4-83 的答案。

4.29　利用 L' = A'B + AB' = A⊕B 得到了图 4-84 所示的方案。

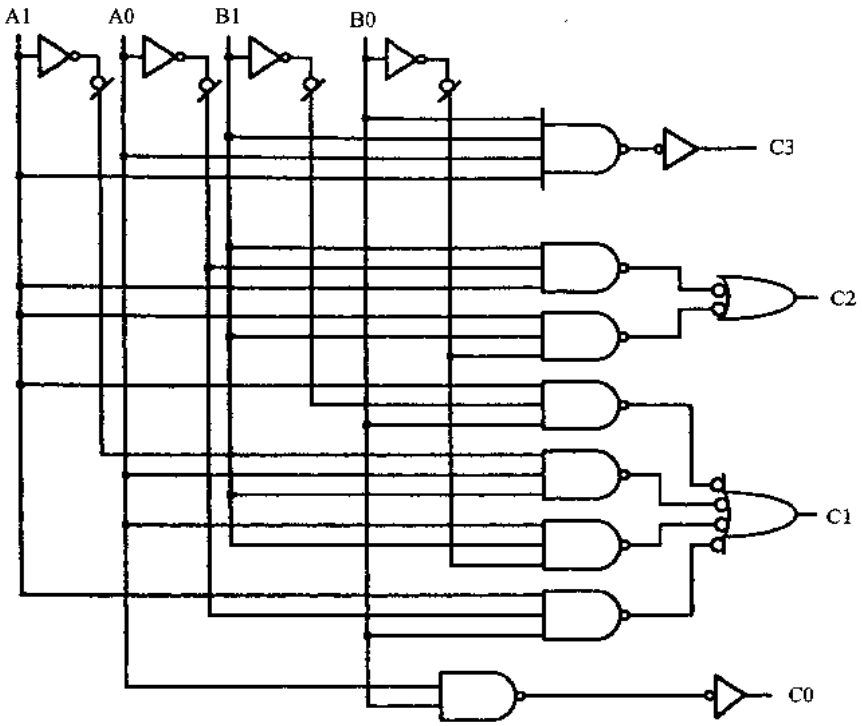
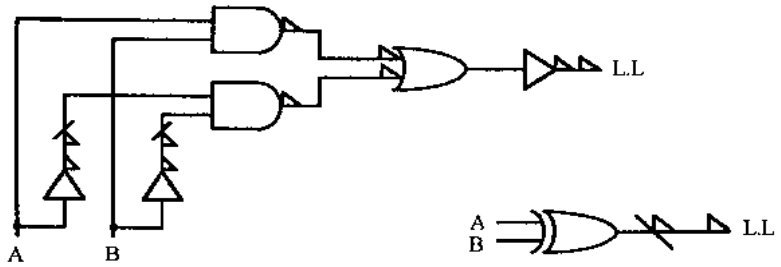
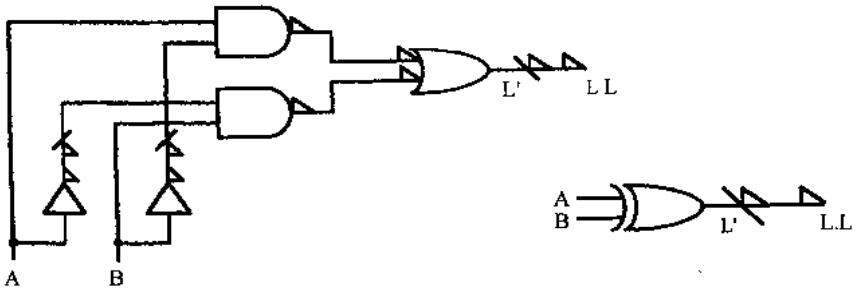


图 4-82　LT 小圆圈规则



与非硬件 $(1\frac{1}{2}$ 芯片$)$　　异或 $(\frac{1}{4}$ 芯片$)$

图 4-83



与非硬件 $(1\frac{1}{4}$ 芯片$)$　　异或 $(\frac{1}{4}$ 芯片$)$

图 4-84

4.30　见图 4-85。芯片数目是 4:1 片二-4 输入与非门,2 片三-3 输入与非门,1 片六倒相器。

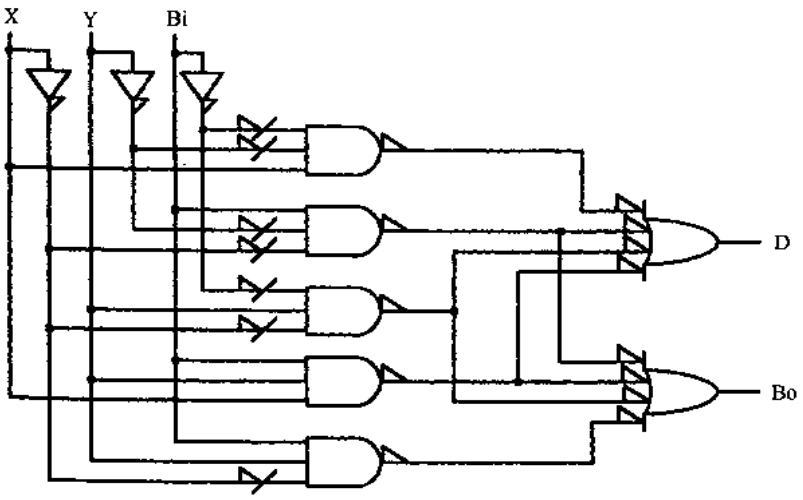


图 4-85

4.31　从第 3.28 题得到 $D = X \oplus Y \oplus B_i$ 和 $B_0 = X'Y + X'B_i + YB_i$。实现如图 4-86。芯片数目为 3：三－3 输入或非，四－2 输入或非，一个六倒相器。

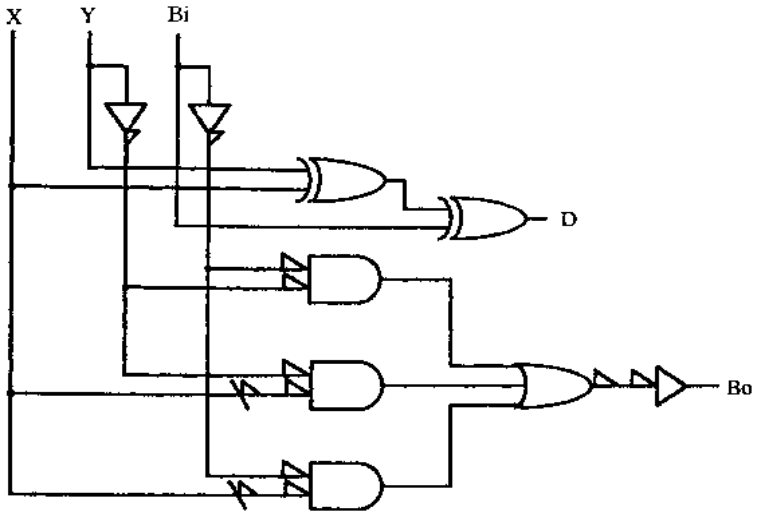


图 4-86

4.32　见图 4-87。

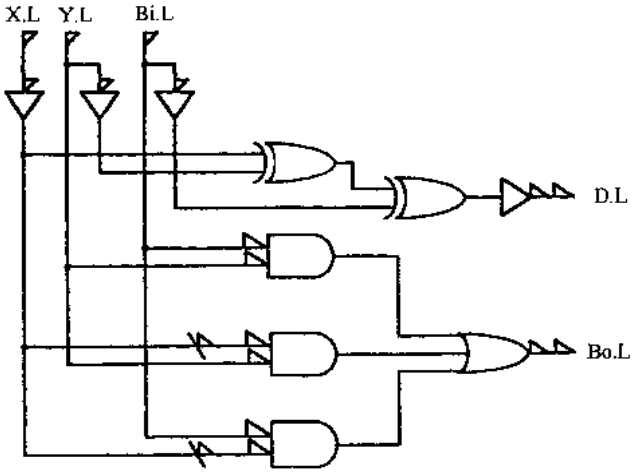
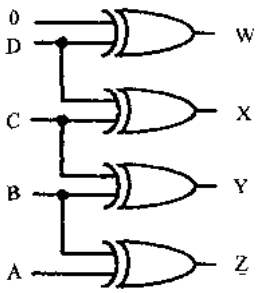


图 4-87

4.33　F = AB + C(A′ + B) = AB + A′C + BC。

4.34　见图 4-88。



W = D = D⊕0
X = CD′ + C′D = C⊕D
Y = B⊕C
Z = A⊕B

图 4-88

因为 4 片异或门封装在 1 个芯片上，这里仅使用 1 个芯片。另一方面，如果使用最初的方程式，需要以下的门：

4 输入与门	8 个门	2 个芯片
8 输入或门	1 个门	1 个芯片
3 输入与门	4 个门	1 个芯片(使用 4 输入或门上的空闲的门作为 1 个)
2 输入与/或门	3 个门	1 个芯片
4 输入或门	1 个门	1 个芯片

因为要求所有的输入信号为直接和求补的形式，可能还需要 1 片六倒相器。因此对于与/或设计将需要 6~7 个芯片。

4.35　F = (B′C + AC + AD)′；实现如图 4-89 所示。

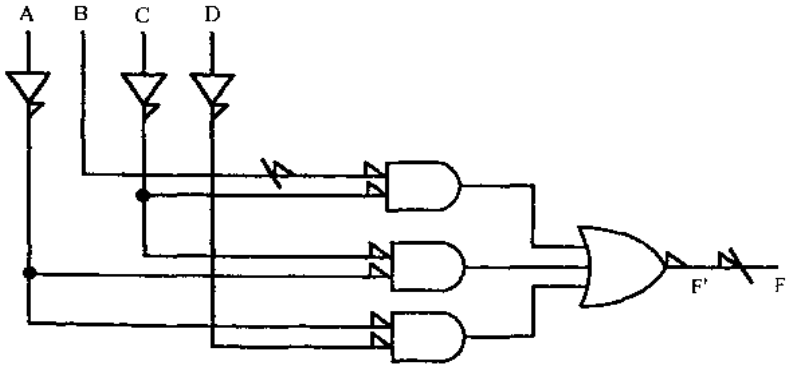


图 4-89

4.36　见图 4-90。

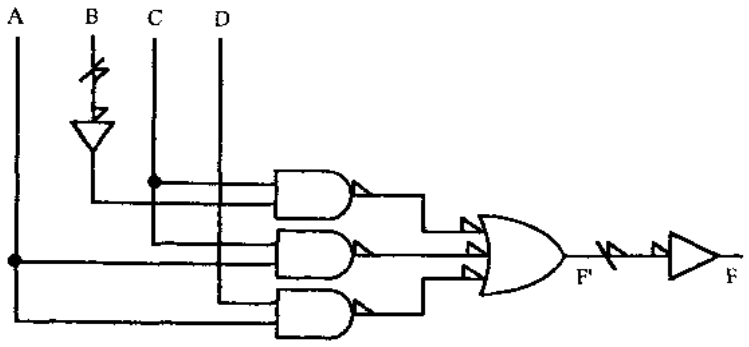
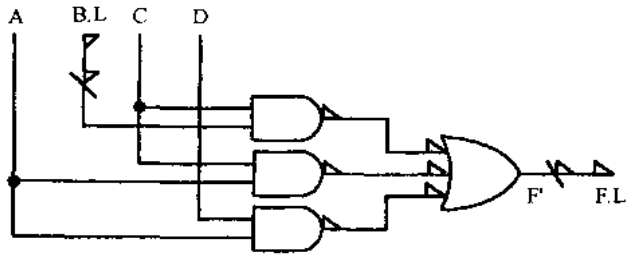


图 4-90

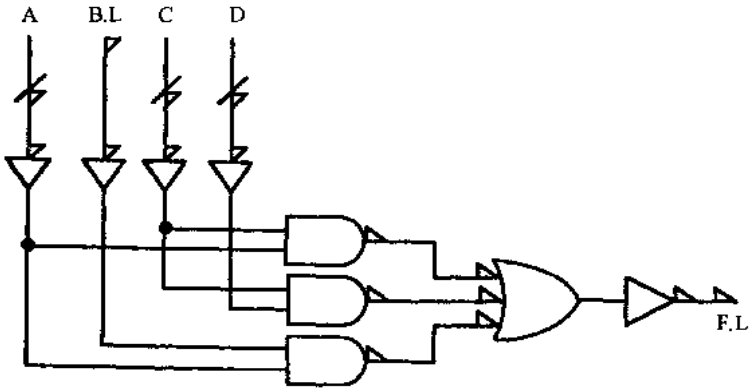
本实现少用 1 个反相器。然而，最长的信号路径通过 4 个门，而在图 4-89 中，关键的时间路径少 1 个门。（参考第 6.11～6.13 题）

4.37　见图 4-91 和 4-92。提示：如何选择方程式形式和/或真值对设计的有效性有极大的影响。



F = (B'C + AC + AD)'

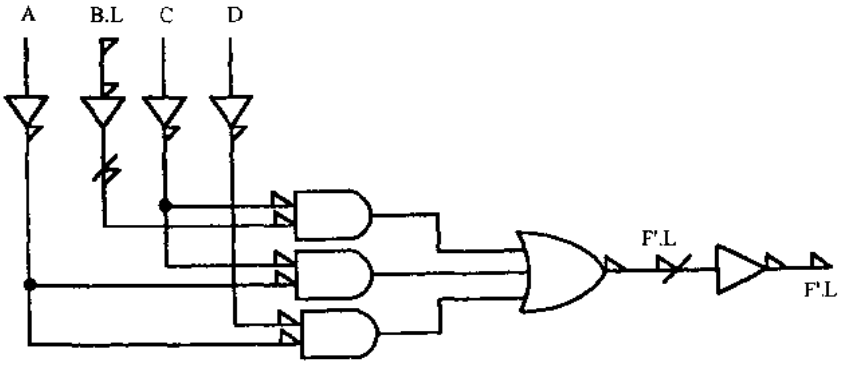
图 4-91



F = A'C' + C'D' + A'B

图 4-92

4.38　见图 4-93 和 4-94。我们明白所需要的反相器的数目和位置取决于所选的卡诺图的包围技术和使用与非还是或非硬件电路。



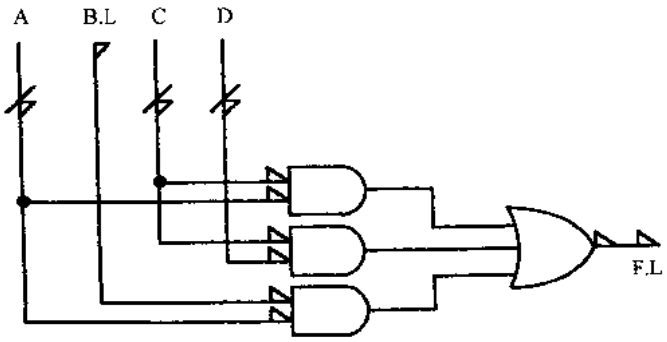
F = (B'C + AC + AD)'

图 4-93

4.39　$X = R(FP)' = \overline{(R' + FP)}$。

4.40　$Y = (R' + FP)$。因为 Y.L = Y'.H，结果变成与上一个问题一致，表明可将两个电路解释成逻辑上相等。对于这个逻辑应用参考第 8.22 题。

4.41　见图 4-95。提示：与上两个问题的正向逻辑形式相比较，从图中获得布尔方程式是相当容易的。



F = A'C' + C'D' + A'B

图 4-94

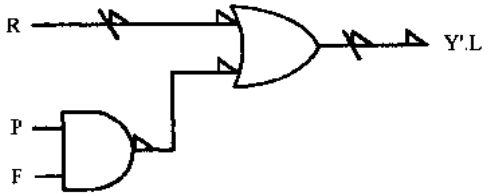


图 4-95

第 5 章　中规模集成电路和大规模集成电路元器件

5.1　引言

随着 IC 技术的发展，有可能将越来越多的门集成在一个硅片上，并且在它们之间有效地创建可靠的互连。这些进步使得用来完成越来越复杂任务的单芯片逻辑电路得以发展。结果和原来由单个门实现的相应的电路相比，芯片需要的空间小得多，而且技术已从小规模集成电路(SSI)发展到中规模集成电路(MSI)，直到大规模集成电路(LSI)；目前我们正在生产超大规模集成电路，它们是如此复杂以至于可以将包括 1 亿个门的整个计算机构建在一块单独的大约指甲盖大小的芯片上。本章将详细讨论 3 种重要的 MSI 和 LSI 器件，多路转换器、译码器和只读存储器以及它们的典型应用。

5.2　多路转换器

多路转换器(MUX)是具有 N 条输入选择线，即 2^N 个数据输入和单条输出线的器件，目前有好几个通用的 MUX 符号，最常用的两种如图 5-1 所示，它可描述为一个 8 数据输入，3 选通输入的形式(参考附录 B)。输入开关(EN)用来启动芯片。如果它为真，所有输入端均正常工作。然而如果它为假，芯片不会接收任何数据。

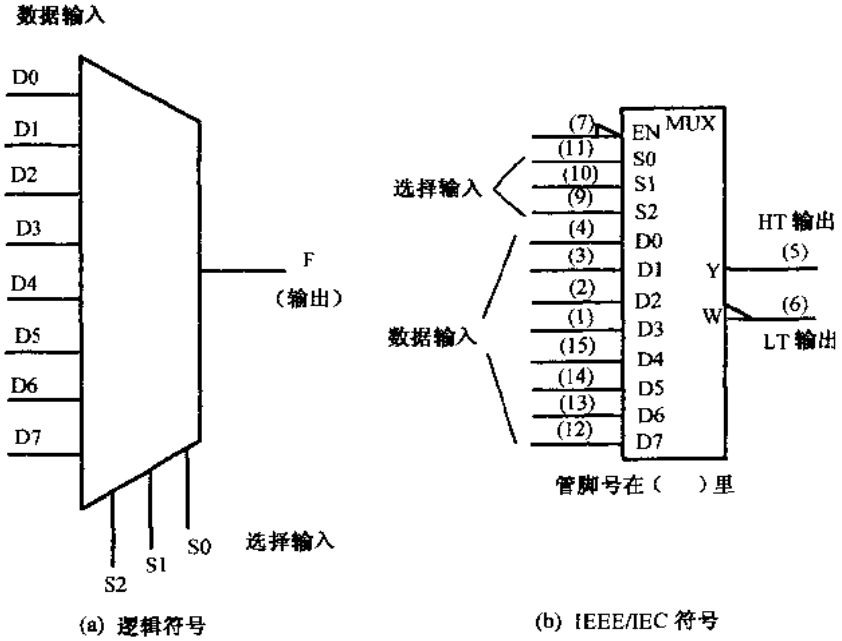


图 5-1

可从两方面理解 MUX。首先，可把它看作是数字机械旋转开关，其次，作为通用逻辑模块(ULM)，可用它来完成通用的组合逻辑函数，同互连的门电路的功能一样。为了理解它的

选择输入端　　表 5.1

S_1	S_0	S_1'	S_0'	G1	G2	G3	G4	F
0	0	1	1	0	0	0	D0	D0
0	1	1	0	0	0	D1	0	D1
1	0	0	1	0	D2	0	0	D2
1	1	0	0	D3	0	0	0	D3

作用，有必要研究器件的内部逻辑。

MUX 的逻辑

两选通，四数据输入的 MUX 的逻辑图如图 5-2 所示。通常，每个选择变量是互补的，因此如果有 N 个选择变量，那么会有 2N 条线连到与门的输入上，每一个与门有 N + 1 个输入，其中之一为数据输入图 5-2 中所示。

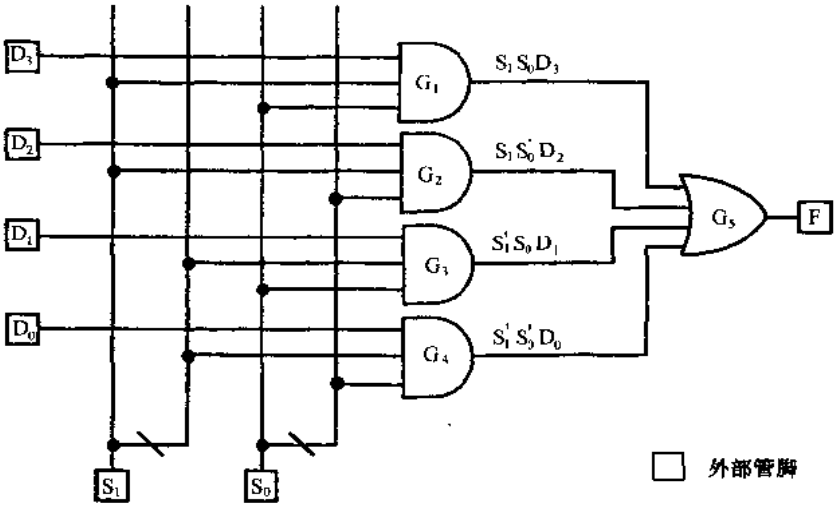


图 5-2

该电路被设计成寻址与门选择变量的惟一组合出现在选择输入端时，输入的数据就被传送到它相关的与门输出端。在两个选择输入的情况下，从表 5.1 可以看出对于每一种选择变量的组合，就会有惟一的选择(数据)输入成为与门的输出。

考虑 $S_1 = 0, S_0 = 1$ 的情况。因为从图 5-2 中可以发现 $G_3 = S_1'S_0D_1$，所以确定了选择输入会使 F 等于 D_1，因此当 $D_1 = 1$ 时 $F = 1$，$D_1 = 0$ 时 $F = 0$。

多路转换器作为开关

可将 MUX 看作是有确定数据输入线，并有选择输入功能的旋转开关，如图 5-3 中描述的

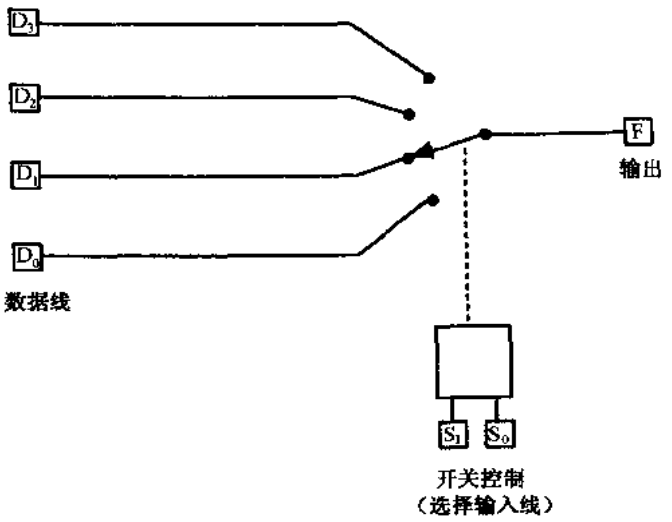


图 5-3

那样。

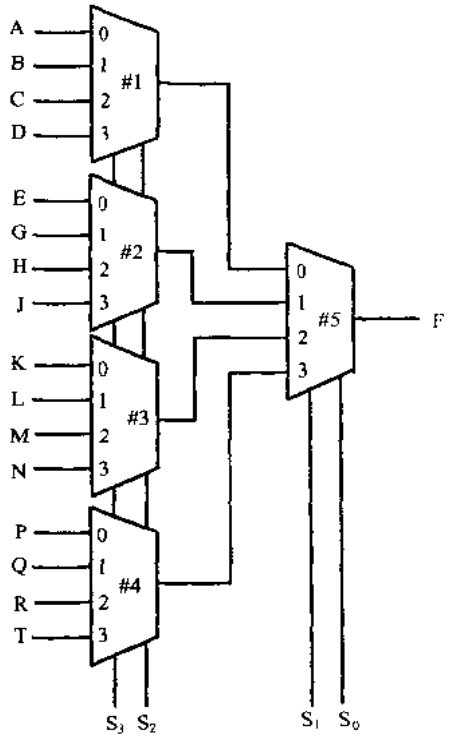


图 5-4

例 5.1　MUX 作为数据开关。如图 5-4 所示的电路有 4 个使用相同选择输入的多路转换器。这些器件的输出为另一个多路转换器的输入，因此，整个电路作为一个具有 16 条数据输入线，4 条选择线的多路转换器。这样的排列称为多路转换器树。如果选择输入是 $S_3S_2S_1S_0 = 1101$，则确定哪一条数据线将被连到 F 上。$S_1S_0 = 01$ 指向第 5 个多路转换器的数据输入 1，通过它将第 5 个多路转换器连在第 2 个多路转换器上。同样，$S_3S_2 = 11$ 指向第 2 个多路转换器的输入 3，信号连在 J 上。因此，惟一的选择输入 1101 将输出 F 连到输入 J 上。

多路转换器作为组合逻辑器件

多路转换器和卡诺图之间有直接的关系，可以通过例子很好地描述。

例 5.2　要求多路转换器的输出 F 通过布尔表达式 F = A′B′C′ + A′BC + ABC 连到它的选择输入 A，B，C 上，给出数据关系。

每个卡诺图方格包括由与方格相关的输入变量的惟一组合决定的逻辑函数 F 的值。如果将这些变量认为是多路转换器的选择变量，那么每个卡诺图方格就包含了与选择变量特定组合相关的多路转换器的输出值。

从图 5-5 中，我们看到卡诺图上显示的给定函数，它的方格用与其二进制地址相等的十进制数标记，为的是将它们与相应的多路转换器数据输入相连。

在卡诺图上，由 ABC = 011 指定的方格包含 1，这表明对于这样的输入组合 F 为真。这个条件对应于给定方程的第 2 项。当选择输入 ABC = 011 时，为了得到多路转换器的输出 1，仅需如图所示将数据输入端 3 连到逻辑 1 上，因为当选择变量组合 ABC = 011 时 F 会被逻辑地连到这个输入上。

用相同的方法确定其他的数据连接。例如，因为卡诺图由 001，010，100，101 和 110 指定的方格都含有 0，所以应当将线 1，2，4，5 和 6 连到逻辑 0 上。

提示：当用卡诺图实现组合逻辑时，不需要卡诺图的化简过程。

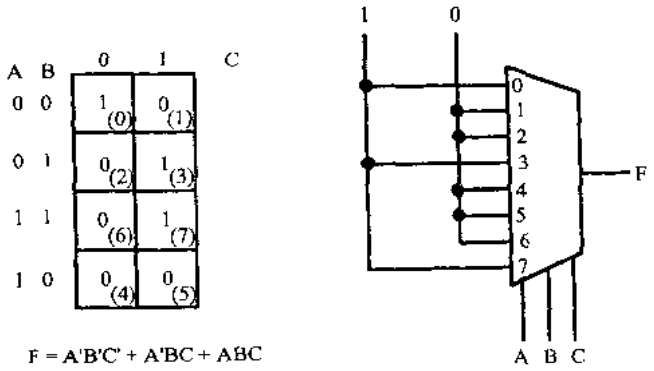


图 5-5

例 5.3　假设卡诺图如图 5-6 所示，如果设计变量比选择输入少使用 1 个，就可以得到更

加有效的多路转换器的逻辑实现。

		0	0	1	1	D
A	B	0	1	1	0	C
0	0	1	1	0	0	
1	0	0	1	0	1	
1	1	1	1	0	1	
0	1	0	1	0	0	

图 5-6

第 1 步：选择设计变量比多路转换器的选择输入线少 1 条。本例中，选择具有 3 条选择输入线的多路转换器。

第 2 步：选择任意的 3 个设计变量作为选择输入。这里从 4 种可能的组合中人为地选择 A，B 和 D。如果这种选择没有其他限制，为了减少错误的可能性，应当尽可能地简化分析，如第 5.5 题所说明的。

第 3 步：考虑在卡诺图和多路转换器上选择不同组合的效果（参考图 5-7）。

ABD＝000。在卡诺图中，参考最顶行左前面的两个位置。因为这两个位置都有 1，不管 C 的值如何，F＝1。在多路转换器上，选择输入组合为 000 使得数据输入 D_0 连到 F 上。因此，应当将最上边的数据线(0)用硬线连到所示的逻辑 1 上。

ABD＝001。在卡诺图上，参考两个右上方位置。不管 C 的值如何，都为 0，F＝0。在多路转换器上，数据输入 D_1 连到 F 上。因此应当将数据线 1 连到逻辑 0 上。

ABD＝010。这一组输入选择在卡诺图上的一对数据不同。因为当 C＝0 时 F＝0；C＝1 时 F＝1，所以可以看出 F＝C。在多路转换器上，数据输入 D_2 将被连到 F 上。因为逻辑要求 F＝C，所以应该将数据线 2 连到输入变量 C 上。

当 ABD＝101 时。卡诺图中的一对数据也不同。但这一次当 C＝1 时 F＝0；C＝0 时 F＝1，因此得到 F 是 C 的逻辑反。在多路转换器上，应当将第 5 条线连到 C′上。

完成连线的多路转换器如图 5-7 所示。可以清楚地看到，这时数据线的输入只是 0，1，C 或 C′。剩下的分析留给学生。

应该弄清楚布尔表达式，真值表，卡诺图，多路转换器之间的关系。并且根据变量数选择合适的多路转换器，能实现任何函数。

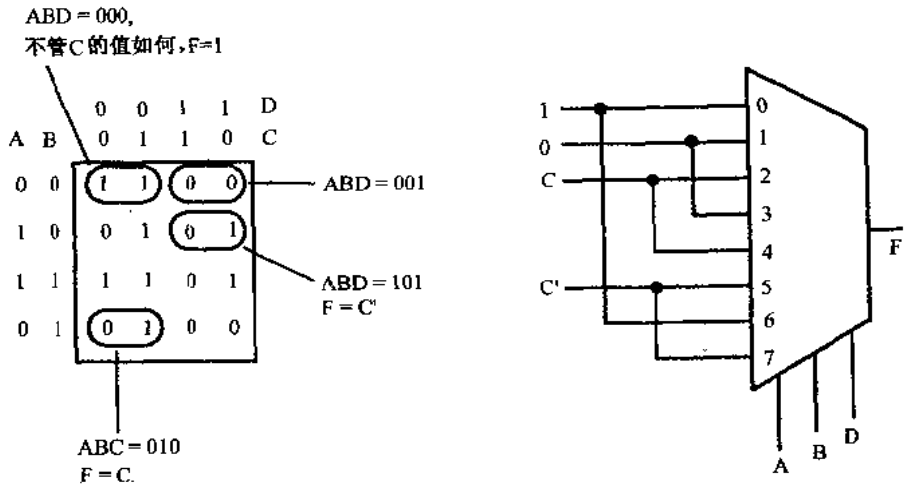


图 5-7

5.3　译码器和多路分配器

有两个密切相关的 MSI 器件，多路分配器与多路转换器功能相反，它根据选择变量的状态将惟一的输入分配到几个数据输出端之一。对于 N 个选择输入端就有 2^N 个数据输出端。除了所有数据输入都连在一起并且没有或门外，逻辑结构与图 5-2 和表 5.1 非常相似。逻辑符号如图 5-8 所示，它表示了多路转换器-多路分配器组合。

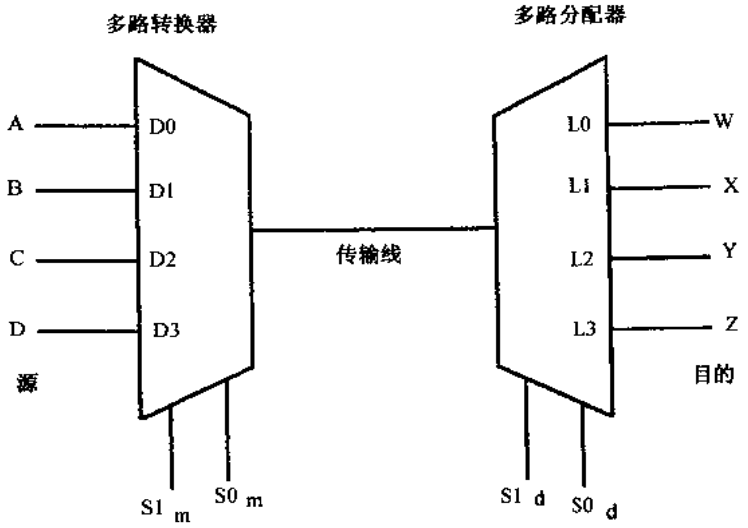


图 5-8

例 5.4　*时分多路转换器*。一个单独的传输装置由 4 个用户共用。这可用图 5-8 所示的多路分配器完成。在多路分配器上，L_0 到 L_3 代表各自的输出(用户线)，而 S_{1d}和 S_{0d}是用来确定输出线的选择输入。

提示：可将图示的安排看成是简单的通讯系统，其中借助于一条传输路径可将任何一个信号源连到任何目的地。

译码器是数据分配器的特殊情况，它将数据输入永久地连到逻辑 1 上，因此，所选的输出将是逻辑 1 的状态而其他输出处于逻辑 0。若所有可能输出都被利用了，那么译码器被称作是"全译码器"。可将全译码器看作是硬件，它表示用选择变量作为真值表的输入行。例如，如果选择是 101，那么会选择真值表的第 5 条线。

如果输出线是连续标示的，很明显可用二进制计数器作为译码器寻址。这里，将二进制数(地址)看作选择输入，由惟一的二进制码选择(寻址)特定的输出线就会出现逻辑 1。

5.4　只读存储器(ROM)

ROM 在图 5-9 中给出。它由寻址译码器(本例中 N = 3)和有一组输入线(每个门有多条输入线)的或门阵列组成。在每个或门的输入端有一条竖线，代表所有的 2^N 个门输入，这里一条线就是寻址线的输出。

输入寻址数据在译码器地址线上就会产生输出。可将这条线连到任何一个或更多或门的输入线。在图中，■表明已连接，○表明没连上。在图 5-9 的例子中，有 8 条译码器输出线和 4 个或门，因此可能的节点总数是 $8 \times 4 = 32$。

如果译码器是由与门阵列组成的，每个与门的输入都被连到惟一的地址组合，并使这个地址与真值表的 1 行相联系，那么译码器对应于真值表的输入侧。对应真值表行的 1 条或多条选择线被连到 1 个或门上，输出变量是 1，这里每 1 个受控的输出变量都有 1 个单独的或门。可将或门连接表解释成真值表的输出。因此，可将 ROM 看成是真值表表 5.2 的硬件，用它来

实现的组合逻辑如图 5-9 所示。

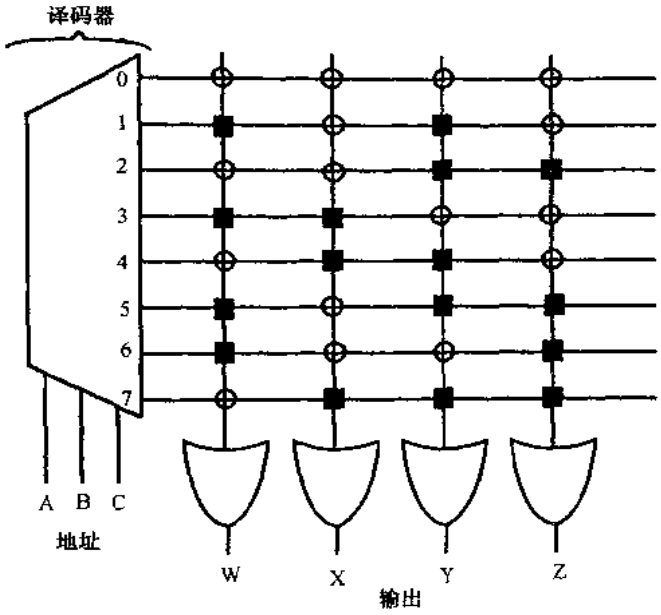


图 5-9

表 5.2

线	A	B	C	W	X	Y	Z
0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0
2	0	1	0	0	0	1	1
3	0	1	1	1	1	0	0
4	1	0	0	0	1	1	0
5	1	0	1	1	0	1	1
6	1	1	0	1	0	0	1
7	1	1	1	0	1	1	1

另一种说法，可将 ROM 看成是存储器件，它以给定地址的结点形式存储数据。将连接结点看成是 1 而没有连接结点是 0。按给定地址线连接的形式被称作地址的“内容”，形成了存放在这里的数据，见图 5-9。如果我们认为 W 是最高位而 Z 是最低位，那么地址 5 包括二进制数 1011 或十六进制数 B。通过正确的设置地址位并读或门的输出可以得到数据。

对 ROM 编程

由译码器和大量或门组成，并且在译码器的输出线和或门的输入线之间没有任何的连结，这样就形成了 ROM。当用户需要一定大小的 ROM 时，用户提供给销售商信息（程序），信息通常是用十六进制数表示地址内容的真值表形式。然后销售商进行正确的金属连接，封装芯片并将它送到用户手中。制造用户可编程的 ROM 也是可能的。这些器件将在第 8 章中讨论。

例 5.5　*二进制到格雷码转换的 ROM*。作为存储器件，可将 ROM 看作是查询表。用真值表二进制输入组作为 ROM 的地址（在表 1.1 中为 $B_3B_2B_1B_0$），将格雷码数存储在图 5-10 所示的相应的地址中。通过输入地址选择能有效的查询真值表中的信息。

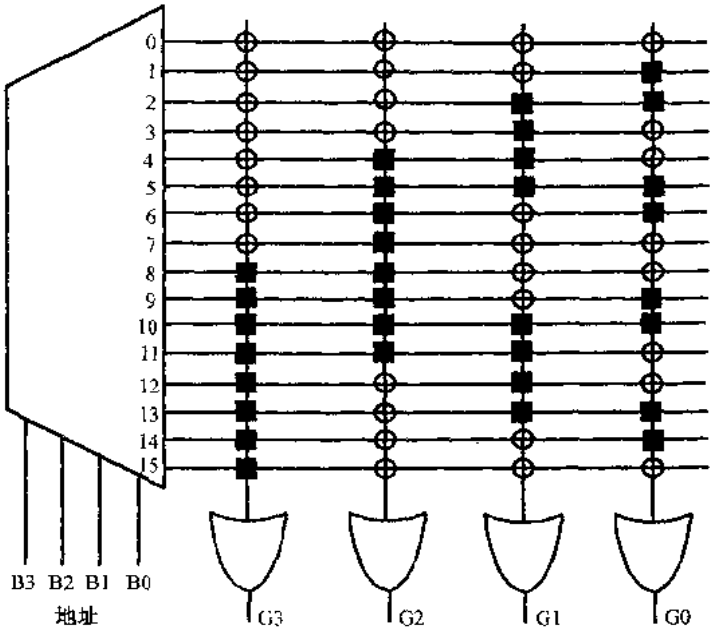


图 5-10

习 题 解 答

5.1　画出有 3 个选择输入的多路转换器逻辑图。

解　☞ 如表 5.1 所示，通过构建真值表的左边确定与门的连接(见表 5.3)。

表 5.3

S_2	S_1	S_0	S_2'	S_1'	S_0'
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	0	0	0

提示：每一行都有 3 个 1 的惟一组合。相应的连接如图 5-11 所示。

5.2　用合适的多路转换器实现 F = C′D + A′BD + A′B′D′ + ABD′。假设 A, B, C 是选择变量。

解　☞ 首先，画出函数的卡诺图，如图 5-12 所示。然后，同例 5.3 一样，将两个分成一组。由变量 A, B 和 C 寻址输入相应的十进制数如图 5-13 所示，这有助于说明过程。最后，同例 5.3 所描述的一样，确定合适的多路转换器数据输入值。结果连接如图 5-14 所示。

5.3　重复 5.2 题，这一次使用 A, C 和 D 作为选择输入端。

解　☞ 修改后的分组如图 5-15 所示，相应的多路转换器连线如图 5-16 所示。

5.4　用多路转换器硬件连接 1 或 0 作为数据输入的方法重复第 5.2 题。

解　☞ 这种情况下，所有变量都作为选择，每一种选择组合代表卡诺图上的一个单独的方格，这样就确定了相应的多路转换器输入。因为多路转换器有 4 个选择输入，所以它是第 5.2 和 5.3 题所使用的多路转换器的两倍，结果在表 5.1 中给出。

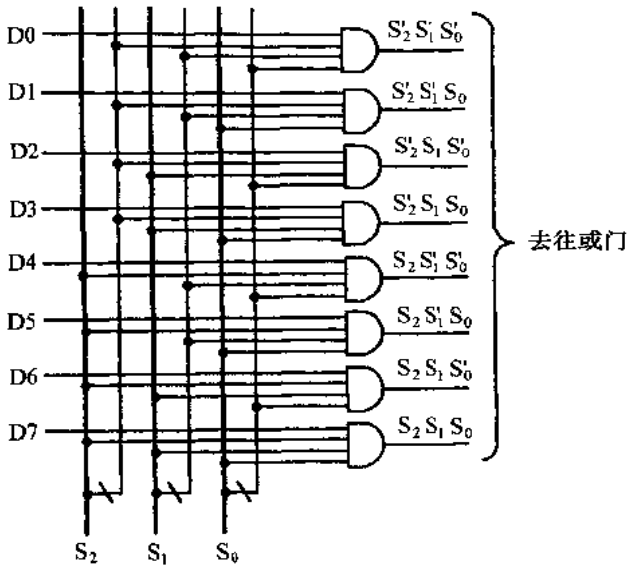


图 5-11

A	B	0	1	1	0	D
		0	0	1	1	C
0	0	1	1	0	1	
0	1	0	1	1	0	
1	1	1	1	0	1	
1	0	0	1	0	0	

图 5-12

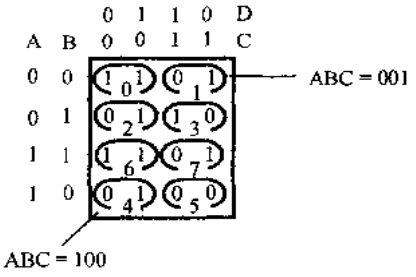


图 5-13

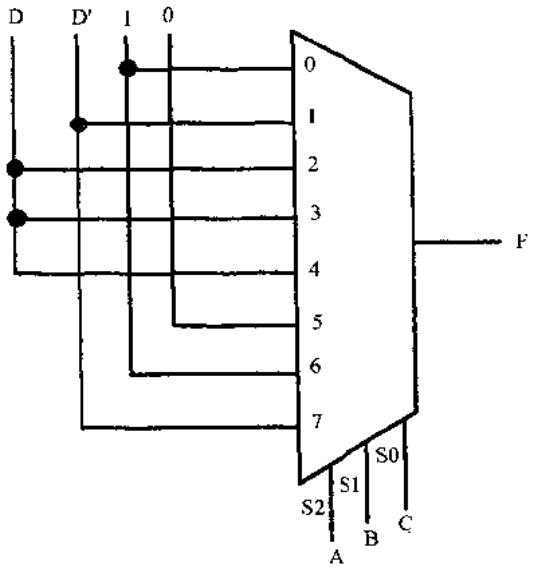


图 5-14

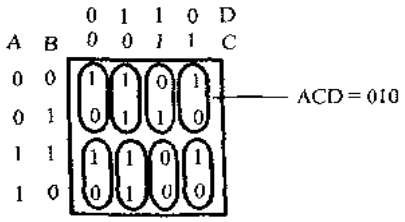


图 5-15

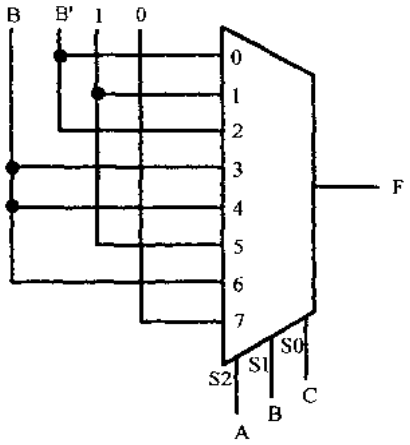


图 5-16

表 5.4

S_3 A	S_2 B	S_1 C	S_0 D	数据线	数据
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	2	1
0	0	1	1	3	0
0	1	0	0	4	0
0	1	0	1	5	1
0	1	1	0	6	0
0	1	1	1	7	1
1	0	0	0	8	0
1	0	0	1	9	1
1	0	1	0	10	0
1	0	1	1	11	0
1	1	0	0	12	1
1	1	0	1	13	1
1	1	1	0	14	1
1	1	1	1	15	0

5.5　画出图 5-17 所示多路转换器的卡诺图，其中 Z 是最小的选择位。

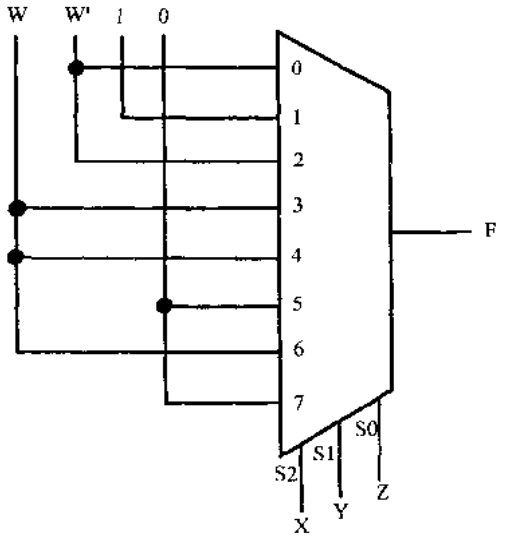


图 5-17

解 对于选择输入是 X=0, Y=1, Z=0 的情况，第 2 条数据线将被选择而且将 W′连到 F 上。因此，当 W=1 时 F 将等于 0；当 W=0 时 F 将等于 1；可表示为 $f_1 = W'X'YZ'$ 的关系式，是 F 的整个布尔表达式中的一项。当 XYZ=001 时，将选择第 1 条数据线，我们看到不管 W 的值是多少，函数都等于 1。这个条件可以表示为 $f_2 = WX'Y'Z + W'X'Y'Z$ 或者，等效成 $F = 1 \cdot X'Y'Z$。对每个输入组合进行相似的分析，去掉所有产生 0 输出的组合，可以完成逻辑函数 F。结果布尔表达式 $F = X'Y'Z + W'X'Y'Z' + W'X'YZ' + WX'YZ + WXY'Z' + WXYZ'$，画出它的卡诺图如图 5-18 所示。

		0	1	1	0	Z
W	X	0	0	1	1	Y
0	0	1	1	0	1	
0	1	0	0	0	0	
1	1	1	0	0	1	
1	0	0	1	1	0	

图 5-18

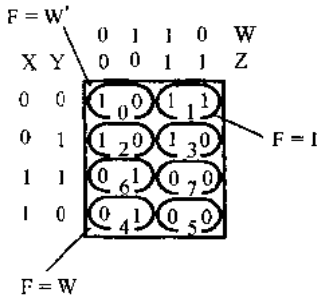


图 5-19

一个简单的方法是画卡诺图时，使 3 个选择变量(本例中为 X 和 Y)中的两个在行和列上是相邻的，第 3 个变量(Z)有两个 1 和两个 0，如图 5-19 所示。通过观察很容易得到各项并填写卡诺图。参考例 5.3 和第 5.2 题。

提示：图 5-18 和图 5-19 的卡诺图看起来不同，实际上它们包含同样的信息。这里要学到的经验是通过变量或表达式重新安排会使问题的难度减小。

5.6 将图 5-20 所示逻辑电路用一个多路转换器来实现。

解 M_1 的输出是 $F_1 = AC'D' + BC'D + CD'$。M_2 的输出是 $F_2 = A'B'C' + A'B + ABD$。那么 $F = F_1 + F_2$，画出如图 5-21 所示的卡诺图。令 ABC 作为选择输入端得到了图 5-22 所示的结果。

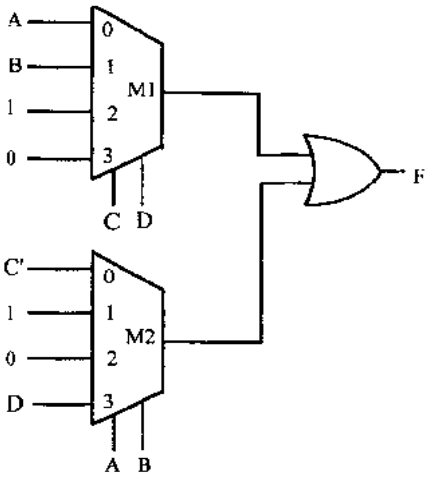


图 5-20

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	1	0	1	
0	1	1	1	1	1	
1	1	1	1	1	1	
1	0	1	0	0	1	

F

图 5-21

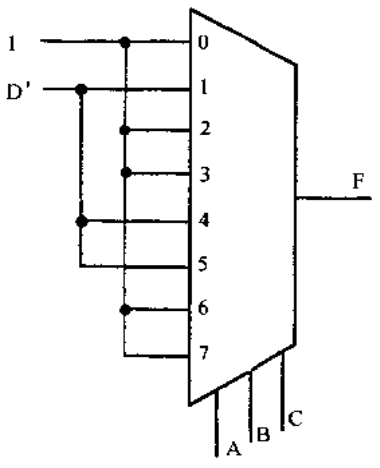


图 5-22

5.7　设计一个 ROM，能完成两个 2 位二进制数相乘的功能。

解　将输入数据称作 A_1A_0 和 B_1B_0 并且创建出必须表示出 4 位输出值的真值表(见表 5.5)。如果我们将输入数据看作是寻址变量并且将每个输出 1 分配到 1 个或门，可从图 5-23 所示真值表的列中直接读取所要求的连接。因为没做连接的圆圈是多余的(5.4 节)，所以将其删去。

表 5.5

A_1	A_0	B_1	B_0	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

表 5.6

输入(格雷码)				输出(二进制数)			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

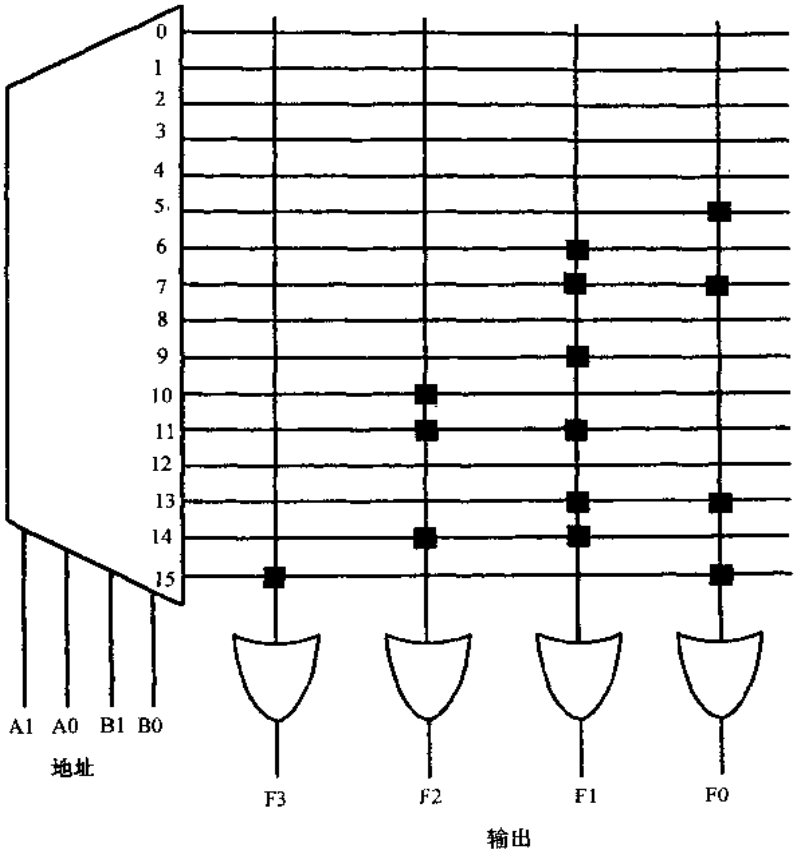


图 5-23

5.8　设计将格雷码转换成二进制码的 ROM 程序。

解　将数字的格雷码序列重新以升序方式排列，放在真值表的输入侧，这在第 2 章中已经讨论过了(参考表 2.1)。现在可将这个序列理解成 ROM 的地址列表，将相对于特定格雷码输入的每个二进制位存储在惟一的 ROM 位置，并且把它作为地址放在选择线上时将有输出。相关的真值表如表 5.6 所示，正确连线的 ROM 在图 5-24 中。

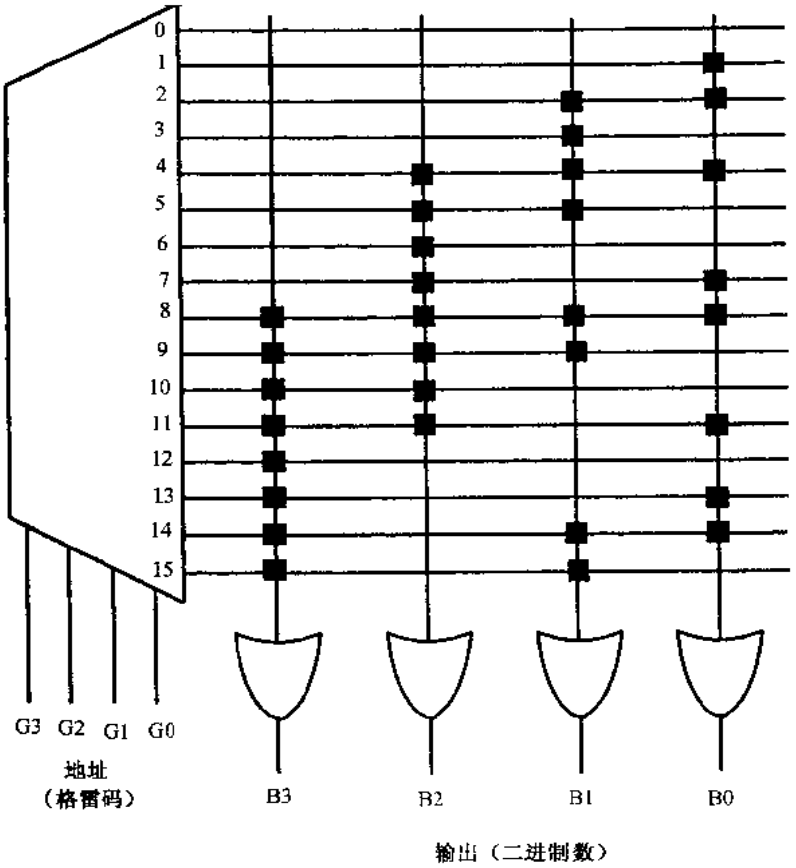


图 5-24

5.9　由图 5-25 所示的硬件图，设计相应的 ROM。

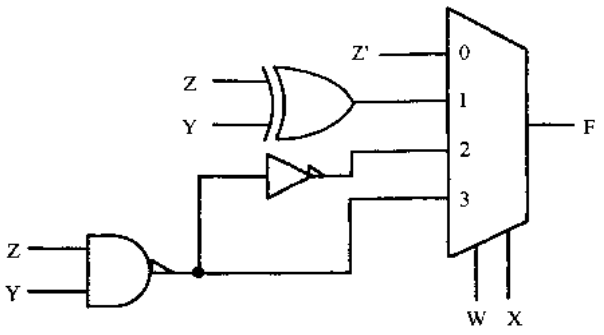


图 5-25

解　提示：有两处半箭头不匹配。因为已知的硬件图，必须在没有添加任何其他硬件的情况下取得平衡。可以通过正确的使用逻辑取反或者斜划线标志(非硬件)来实现。使划线标志在插入点处重新定义真的极性(LT 或 HT)。这个过程可以通过复习 4.3 节的关于电压反相器的定义和与逻辑取

反过程的关系来理解。结果平衡的电路图如图 5-26 所示，其中用替代的小圆圈规则而不是半箭头。

一旦得到了平衡，就去掉所有的小圆圈和硬件反相器（不是斜划线）来阐明内在的逻辑图，从而通过观察得到布尔表达式：

$$F = W'X'Z' + W'X(Y \oplus Z) + WX'YZ + WX(YZ)'$$
$$= W'X'Z' + W'XYZ' + W'XY'Z + WX'YZ + WXY' + WXZ'$$

结果卡诺图和 ROM 如图 5-27 所示。

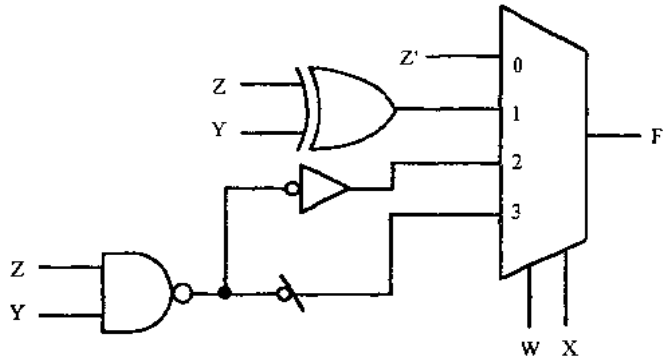


图 5-26

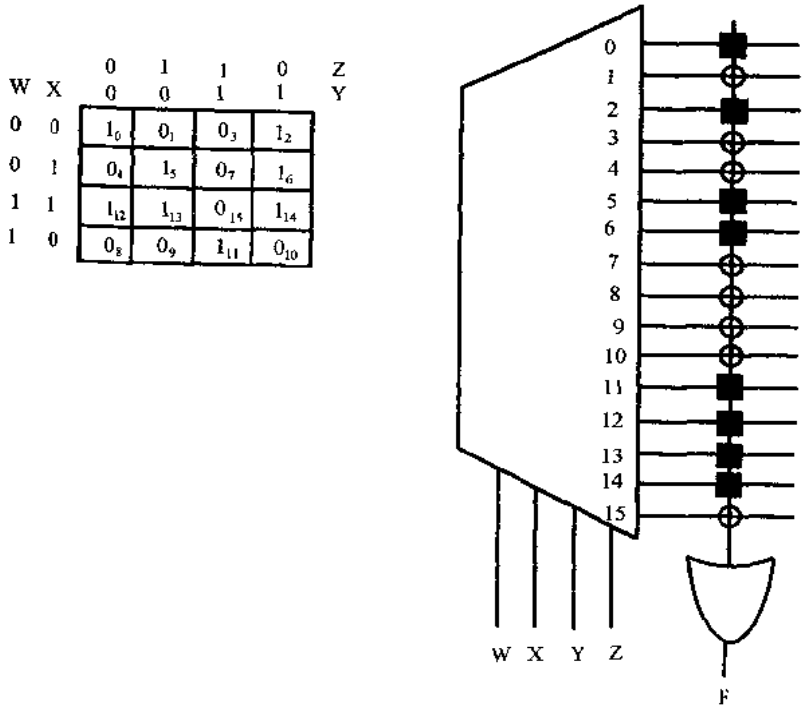


图 5-27

5.10　用一个多路转换器实现图 5-28 所示的逻辑。

解　☞ 从逻辑图可知布尔方程式是

$$F = XZ' + WYZ + W'X'Z + W'YZ' + WXY$$

画出卡诺图得到图 5-29，在图 5-29 中发现，WXY 项是多余的，因为所有的 1 都涉及到了却没有这一项。从卡诺图得到表 5.7，从表中可以做出多路转换器的连接，如图 5-30 所示。

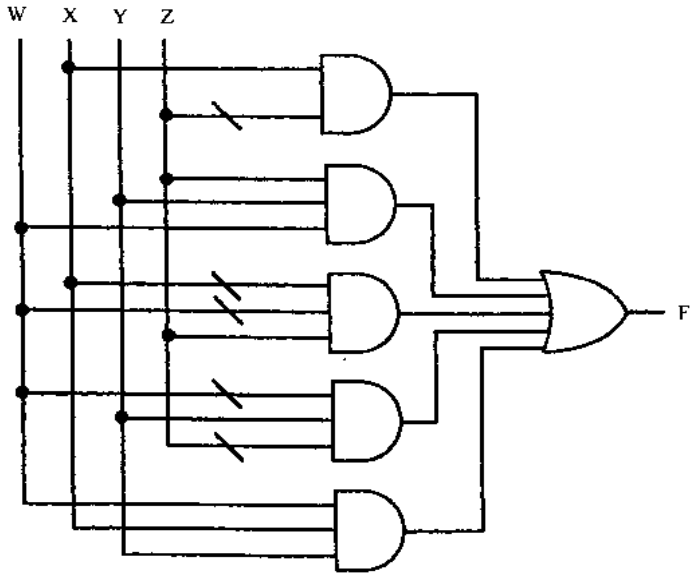


图 5-28

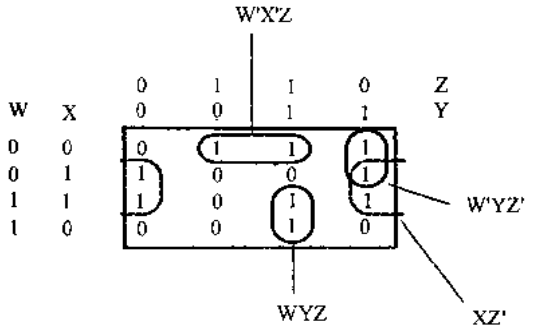


图 5-29

表 5.7

	W	X	Y	F	
0	0	0	0	Z	当 Z=0 时 F=0 当 Z=1 时 F=1
1	0	0	1	1	
2	0	1	0	Z'	当 Z=0 时 F=1 当 Z=1 时 F=0
3	0	1	1	Z'	
4	1	0	0	0	
5	1	0	1	Z	
6	1	1	0	Z'	
7	1	1	1	1	F=1 与 Z 无关

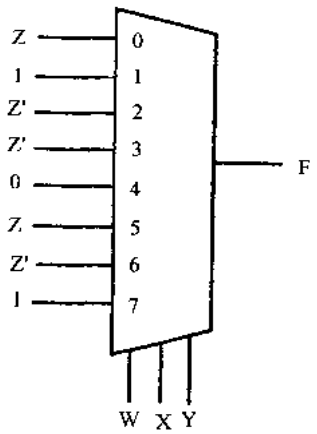


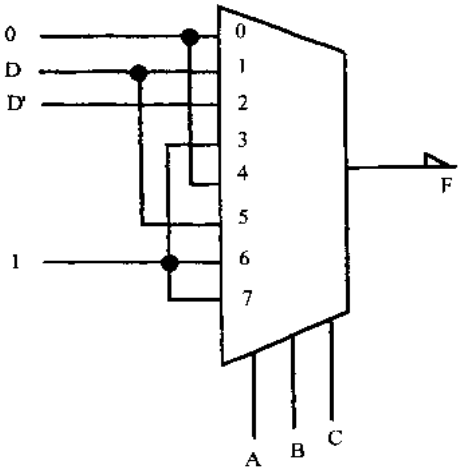
图 5-30

5.11　假设多路转换器电路如图 5-31 所示，用与非门完成相应的逻辑。

解　现在，学生应该有足够的技巧直接从多路转换器中填写卡诺图。然而，通过构建与表 5.7

相似的表有助于这个过程。

目的是用与非门实现，进行简化来减少门的数目是合情理的。用卡诺图包围 1 的方法如图 5-32 所示，并且函数是 F = AB + CD + BD′。与非逻辑图的结果如图 5-33 所示。



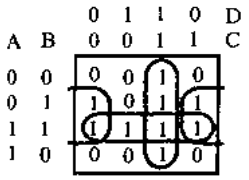


图 5-32

图 5-31

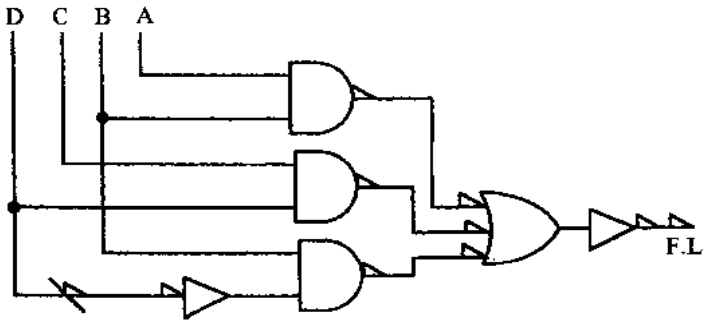


图 5-33

5.12　用多路转换器实现 4 位数字的奇偶校验；也就是，如果数字中 1 的个数是奇数电路应输出真。



图 5-34

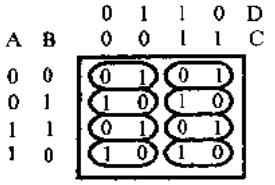


图 5-35

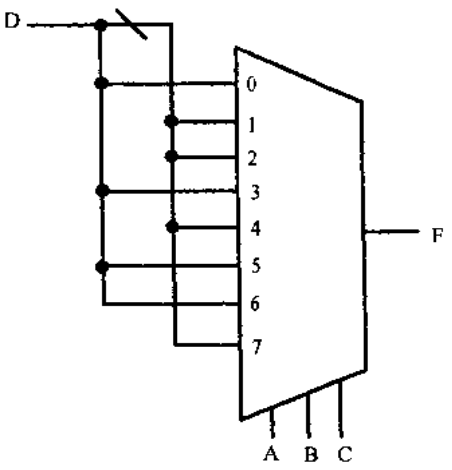


图 5-36

解 这将会得到 4 个独立变量和 1 个非独立变量(图 5-34)。画出函数的卡诺图，并象题 5.5 那样对最小项分组，得到图 5-35，用 A, B, C 作为多路转换器的选择输入，得出如图 5-36 所示的结果。

5.13 假设硬件电路如图 5-37 所示，用单个多路转换器实现相应的逻辑。

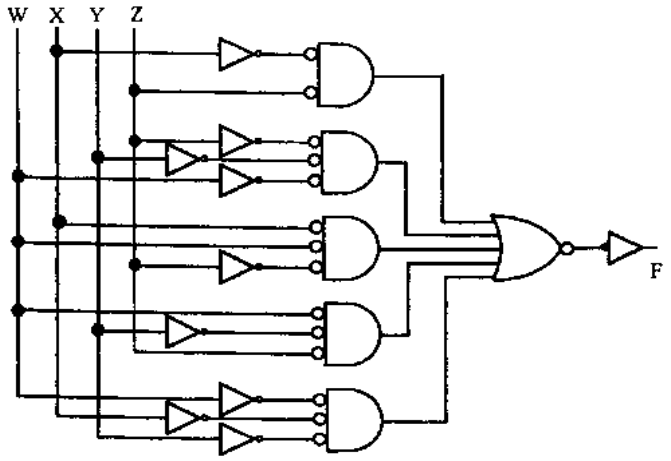


图 5-37

解 如果我们首先通过添加逻辑取反来平衡小圆圈，可以很容易的解出逻辑，如图 5-38 所示。然后去掉所有小圆圈和硬件反相器(留下斜划线标志不动)，可从剩余的逻辑图中直接读出布尔方程式为

$$F = XZ' + WYZ + W'X'Z + W'YZ' + WXY$$

画出方程的卡诺图，得到图 5-39。

最后，像题 5.5 一样给卡诺图中的项分组后并使用 W, X 和 Y 作为选择变量，得到连好线的多路转换器，如图 5-40 所示。

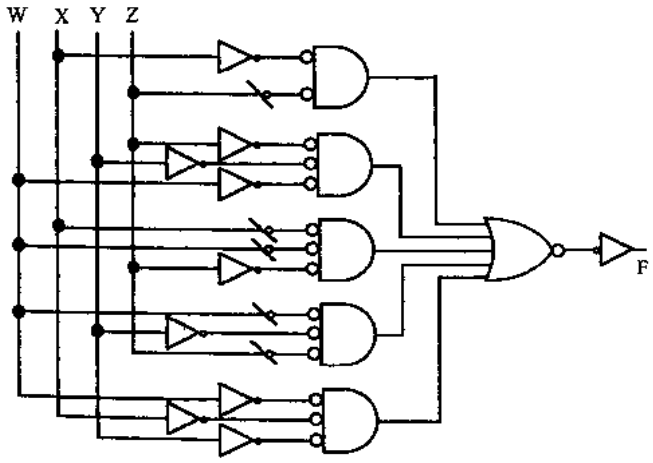


图 5-38

		0	1	1	0	Z
W	X	0	0	1	1	Y
0	0	0	1	1	1	
0	1	1	0	0	1	
1	1	1	0	1	1	
1	0	0	0	1	0	

图 5-39

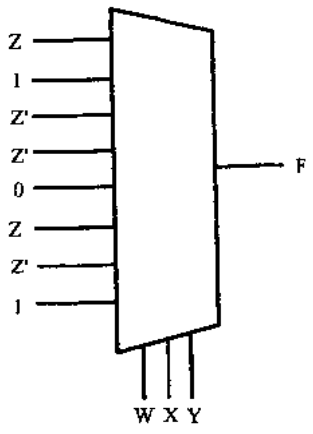


图 5-40

5.14　确定对应于图 5-41 所示硬件电路图的布尔表达式。

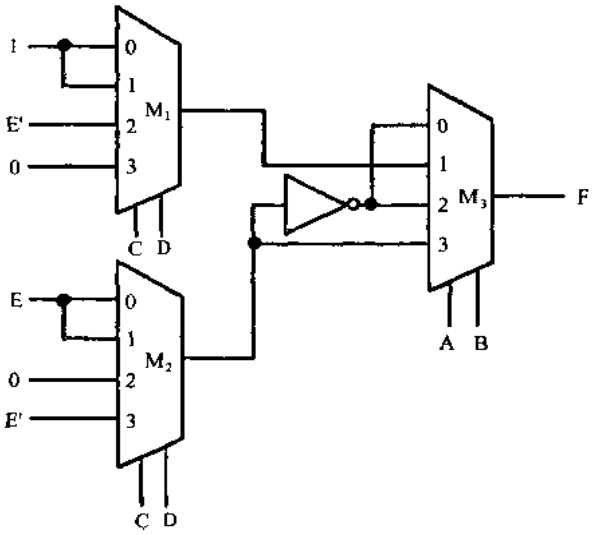


图 5-41

解　因为在多路转换器的 M_2 和 M_3 之间有 1 个硬件反相器，一定要通过添加逻辑非来平衡小圆圈，如图 5-42 所示。假设所有指定的变量都是 HT。考虑图 5-41 中的 M_3：

$$A'B' \rightarrow F = M_2'$$
$$A'B \rightarrow F = M_1$$
$$AB' \rightarrow F = M_2'$$
$$AB \rightarrow F = M_2$$

并且，
$$M_2 = C'D'E + C'DE + CDE'$$

可以画出卡诺图(图 5-43)并用包围 1 的方式化简，从而得到

$$M_2 = C'E + CDE' \tag{5.1}$$

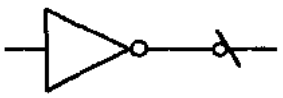


图 5-42

C D	0	1 E
0 0	0	1
0 1	0	1
1 1	1	0
1 0	0	0

M2

图 5-43

用包围 0 的方法得到

$$M_2' = C'E' + CD' + CE \tag{5.2}$$

同样地，

$$M_1 = C'D' + C'D + CD'E'$$
$$= C' + CD'E' \tag{5.3}$$

现在可以构造出函数 F

$$F = A'B'M_2' + A'BM_1 + AB'M_2' + ABM_2$$
$$= B'M_2' + A'BM_1 + ABM_2$$

将方程式(5.1)，(5.2)和(5.3)替代进(5.4)中，我们得到

$$F = B'C'E' + B'CD' + B'CE + A'BC' + A'BCD'E' + ABC'E + ABCDE'$$

将其画在卡诺图 5-44 中。用包围 1 的方法实现简化，产生

$$F = A'C'E' + CD'E' + B'C'E' + BC'E + B'CE$$

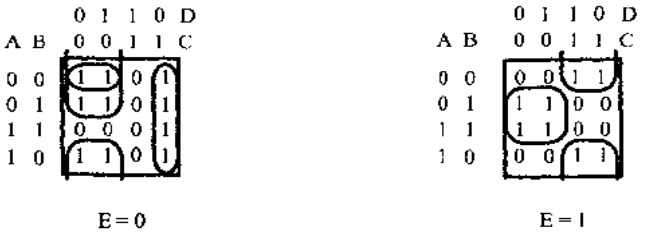


图 5-44

5.15　求出图 5-45 所示 ROM 的每一个输出的简化布尔表达式。

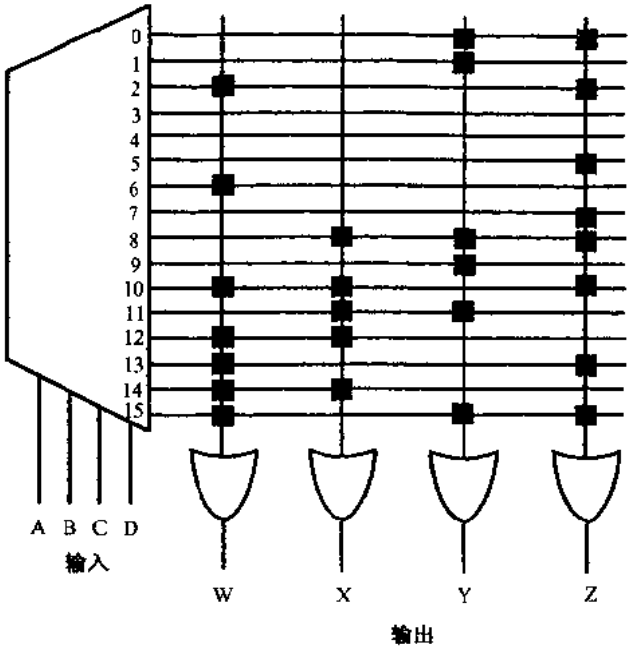


图 5-45

解　从 ROM 的连接图中可直接画出函数的卡诺图(图 5-46)。

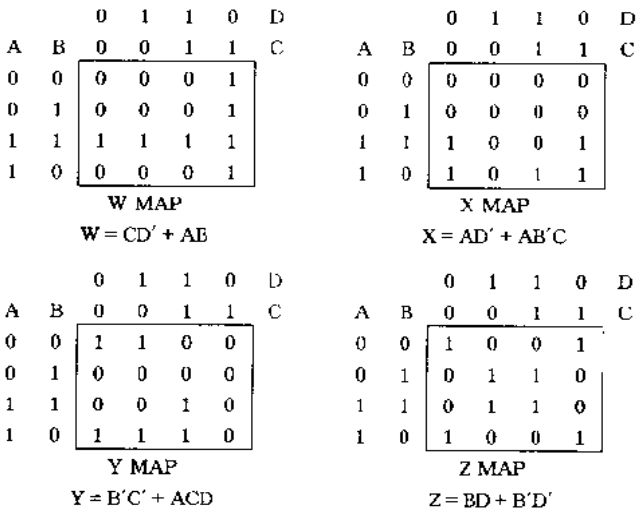


图 5-46

5.16　已知图 5-47 所示的 ROM，确定十六进制数 E1 存放的地址。提示：可选择使用实心圆圈表示连接。

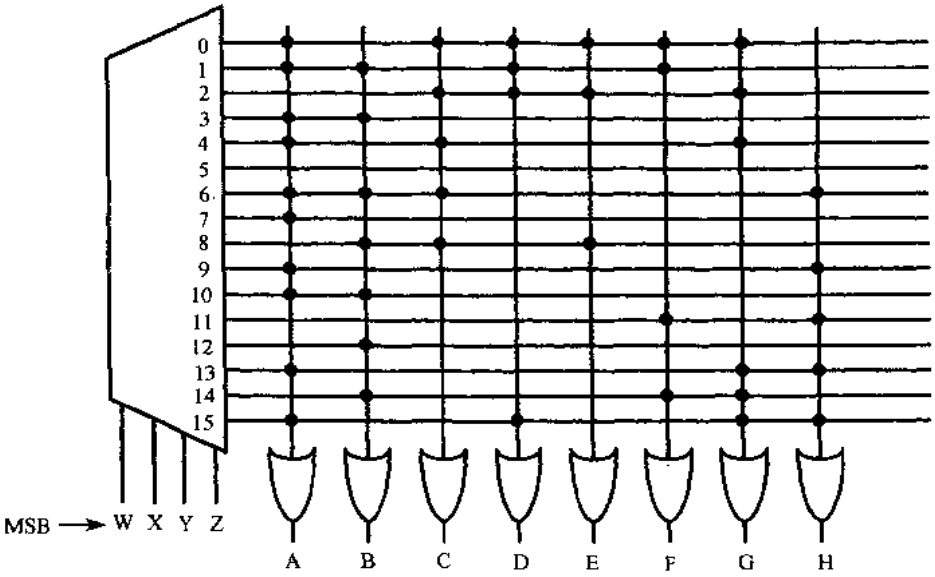


图 5-47

解 ☞ 4 个一组，得到 $ E1 = (1110)(0001)，可看出来它被存储在第 6 行。二进制地址是 WXYZ = 0110。

补 充 习 题

5.17　假设多路转换器电路如图 5-48 所示，用小规模集成电路与非门实现相应的逻辑电路。

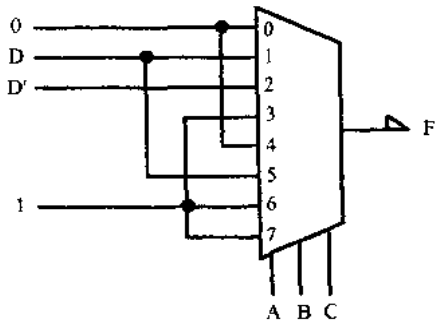


图 5-48

5.18　在 1 片 3 选择的多路转换器上实现如图 5-49 所示的逻辑。

5.19　用与非硬件实现图 5-50 所示多路转换器的逻辑，并尽可能减少门的数量。假设所有输入变量都可用作互补和逻辑取反形式。

5.20　实现第 3.26 题讨论的汉明检错纠错码的 P_3 位的奇偶校验逻辑。要求用没有化简的布尔方程式卡诺图和一个 3 选择输入的多路转换器。

5.21　用布尔方程式化简的卡诺图和一个 2 选择输入的多路转换器重复第 5.20 题。

5.22　用多路转换器实现第 3.25 题的 2 位乘法器的解决方案。与第 4.27 题的小规模集成电路实现方案比较芯片的数量。

5.23　用多路转换器实现第 2.27 题的二进制数减法器的解决方案。与第 4.30 题的小规模集成电路实现方案比较芯片的数量。

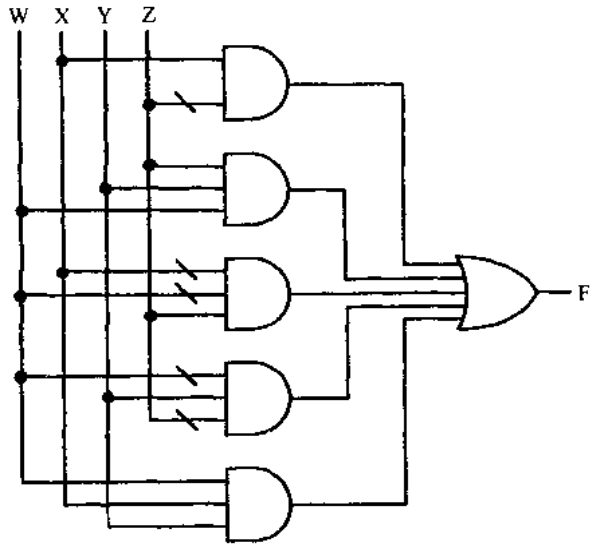


图 5-49

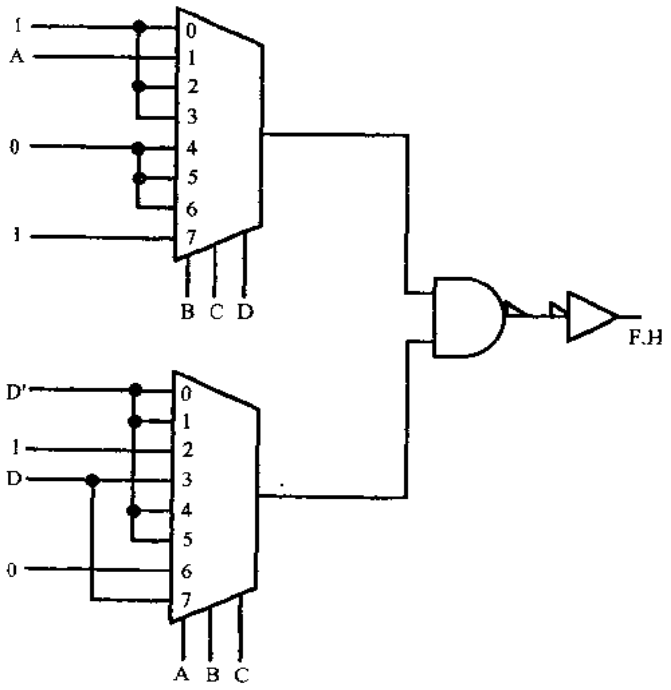


图 5-50

5.24　用 3 选择输入的多路转换器实现第 5.20 题的二进制数减法器的解决方案。与第 5.23 题的解决方案比较结果。

5.25　图 5-51 中，A 和 F 是最高位，把电路看作是存储器件，存储在 1011 和 0101 处的二进制数是什么？

5.26　将图 5-51 看作是逻辑器件，画出变量 J 的卡诺图并简化逻辑。

5.27　假设连好线的 ROM 如图 5-52 所示，用一个 3 选择输入的多路转换器实现输出 F 的逻辑，用 W，X 和 Z 作为选择变量，W 作为最高位。

5.28　设计 3 选择输入的多路转换器电路，当且仅当它的 4 个输入为 1 的数目为偶数时，它的输出为 1。

5.29　用 ROM 实现 4 位数字的奇偶校验，如果数字中 1 的个数为奇数时，电路输出 1。假设二进制数是 A，B，C 和 D，A 是最高位。

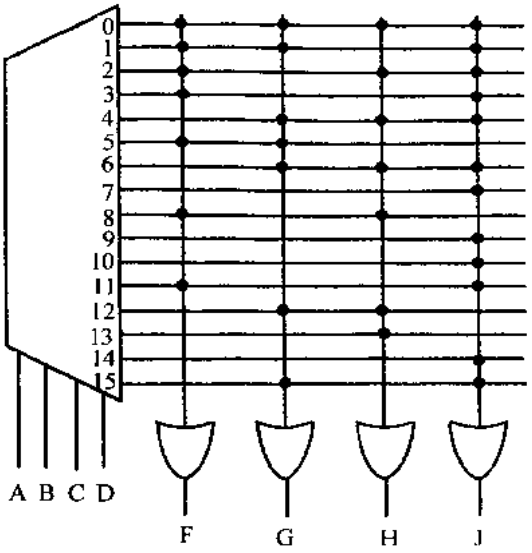


图 5-51

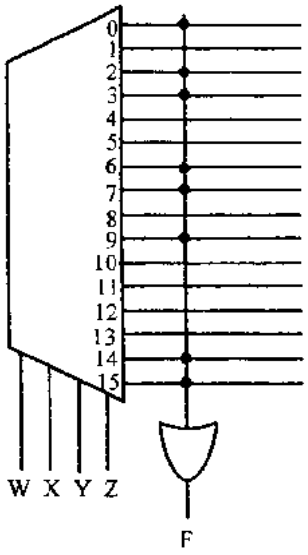


图 5-52

5.30　用 ROM 实现第 2.2 题的 7 段显示驱动器。

5.31　将图 5-53 所示的 ROM 看作是存储器件，确定所有地址的内容。

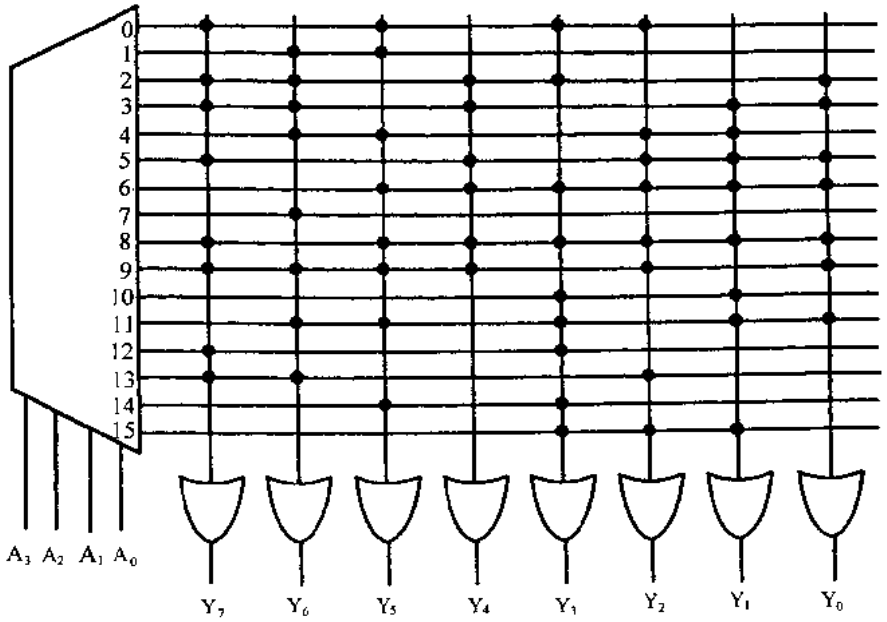


图 5-53

5.32　将图 5-53 所示的 ROM 看作是一个装置，如图 5-54 所示，对于所示的输入给出系统的输出(Z_1, Z_2)。

5.33　参考第 2.2 题，用 3 选择输入的多路转换器实现 7 段译码段的驱动逻辑。

5.34　在 1 个芯片上的 8 输入多路转换器。如果 1 个多路转换器花 $0.25，1 个 ROM 为 $0.125，安装和检测的其他芯片花了 $0.35，比较整个 7 段显示驱动器的多路转换器实现和 ROM 实现的花费。

5.35　考虑图 5-55 所示的开关网络，施加什么选择输入信号将 b 和 N 连起来？

5.36　考虑图 5-55 所示的开关网络，假设当选择信号为

$$S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8 = 01110101$$

时将做什么样的连接？

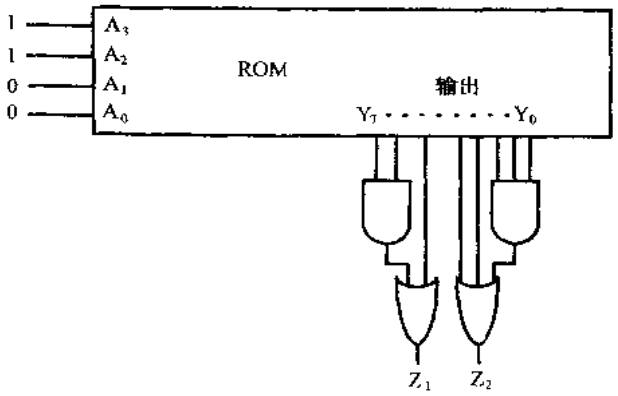


图 5-54

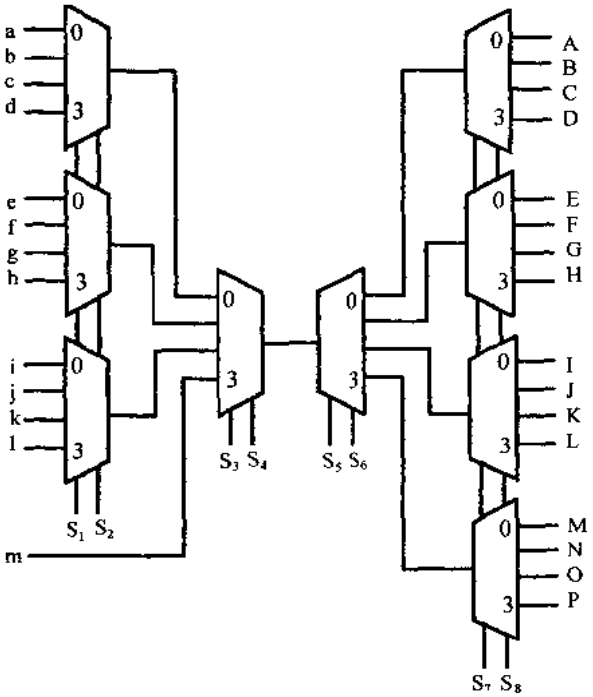


图 5-55

5.37　对于图 5-55 所示的开关网络，估计用单个小规模集成电路门实现它所需要的芯片数量。将其与多路转换器的实现相比较，提示：两个 4 输入的多路转换器封装在 1 块芯片上，参考图 5-2 和第 5.3 节。

5.38　对于第 5.37 的两个竞争设计，每个门芯片的花费为 $0.20，每个双多路转换器或多路分配器的花费是 $0.80。如果安装测试每 1 个芯片花掉 $0.35，并需要 1 in^2 的面积，比较两种实现中每 1 个需要的花费和面积，为了在价格上相等，确定多路转换器的价钱。

5.39　考虑用 1 片 ROM 实现第 4.27 题的 2 位二进制数乘法器，1 个门电路芯片花掉 $0.20，其余的安装测试花了 $0.35。如果安装、测试 ROM 花费 $0.85，那么为了和相关的单个门电路设计的花费持平，可标出的最大价格是多少？

5.40　一种便宜的商用 ROM 可以使用 8 条地址线和 8 条输出线，那么有多少种可能的连接？

补充习题答案

5.17　见图 5-56。

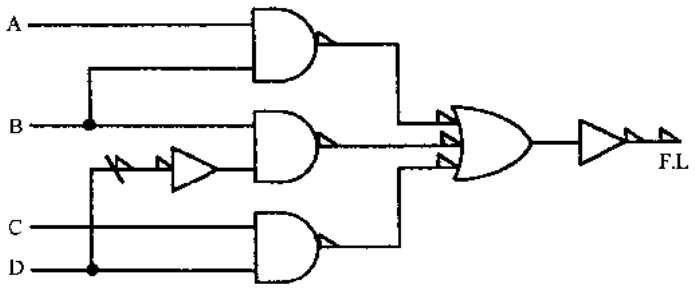


图 5-56

5.18　见图 5-57。

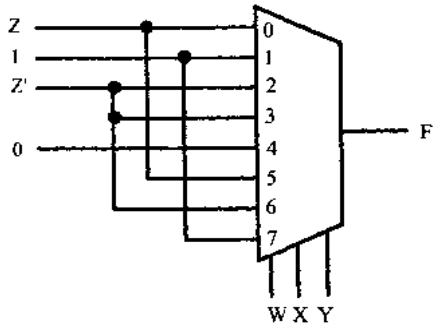


图 5-57

5.19　见图 5-58。提示：变量 A 对输出结果没有影响。

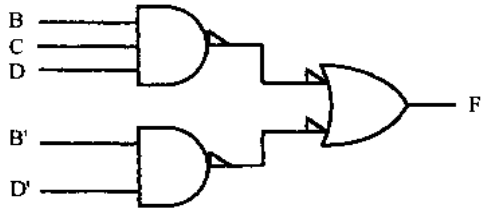


图 5-58

5.20　从第 3.26 题中得到

$$P_3 = D_4'D_3'D_2'D_1' + D_4'D_3'D_2'D_1 + D_4'D_3D_2D_1' + D_4'D_3D_2D_1 + D_4D_3'D_2'D_1' + D_4D_3'D_2D_1 + D_4D_3D_2'D_1' + D_4D_3D_2'D_1$$

见图 5-59。

D_4	D_3	0 1 1 0 (D_1) / 0 0 1 1 (D_2)
0	0	1 1 0 0
0	1	0 0 1 1
1	1	1 1 0 0
1	0	0 0 1 1

P_3

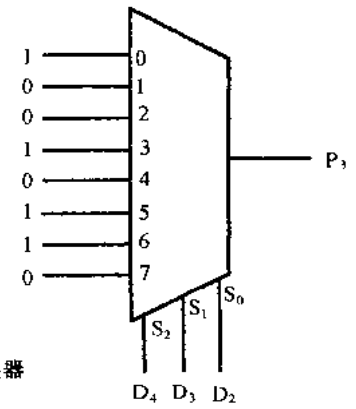


图 5-59

5.21　见图 5-60。

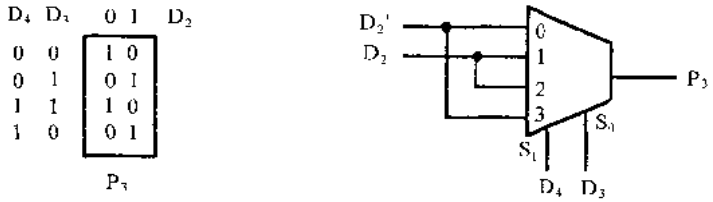


图 5-60

5.22　见图 5-61。提示：与 4.27 题的优化的 SSI 情况使用 5 个芯片相比，仅需使用 4 个。

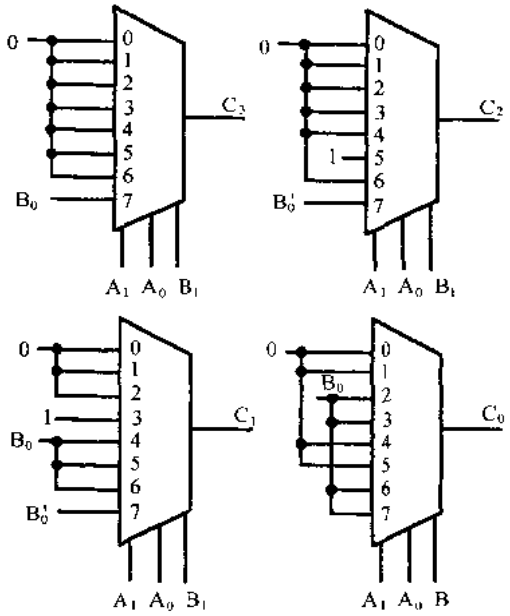


图 5-61

5.23　见图 5-62。

X	Y	B_1	D	B_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

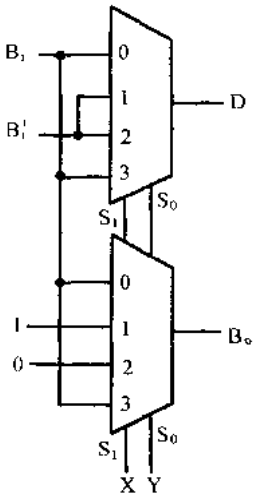


图 5-62

5.24　解决方案如图 5-63 所示，因为三选择输入的多路转换器封装在 1 个芯片上。所以需要两片。不需要

反相器，然而用这种方法可使器件的布局变得容易。

X	Y	B_i	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

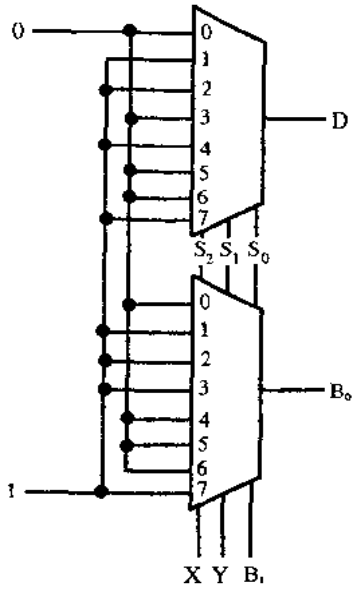


图 5-63

5.25　1001，1100。

5.26　见图 5-64。$J = C + B'D + A'D'$。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	1	1	1	
0	1	1	0	1	1	
1	1	0	0	1	1	
1	0	0	1	1	1	

图 5-64

5.27　见图 5-65。

5.28　见图 5-66。

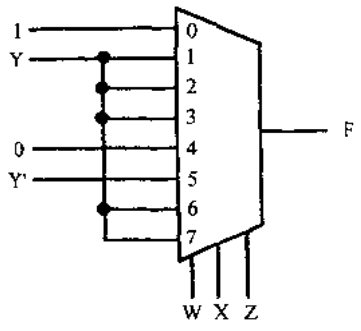


图 5-65

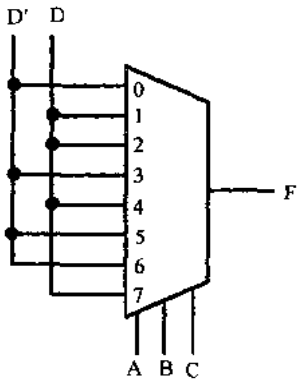


图 5-66

5.29　见图 5-67。

5.30　见图 5-68。

5.31　见表 5.8。

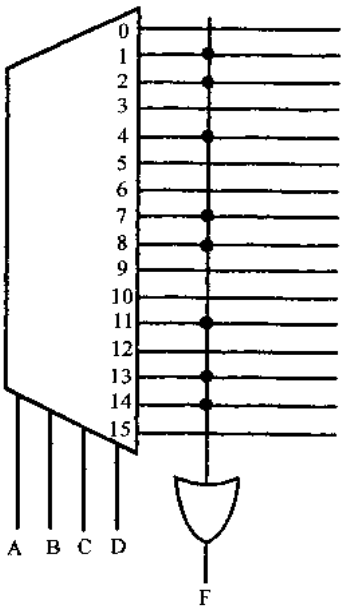


图 5-67

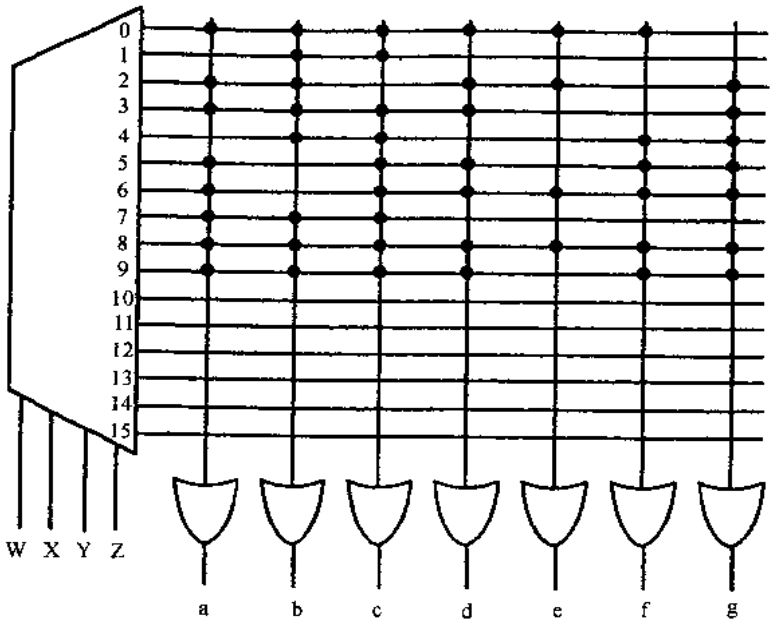


图 5-68

表 5.8

地址	内容	地址	内容
0	AC	8	BF
1	60	9	F5
2	D9	10	0A
3	D3	11	6B
4	66	12	88
5	97	13	C4
6	3F	14	28
7	40	15	0E

5.32　$Z_1 = 0, Z_2 = 1$。

5.33　见图 5-69。

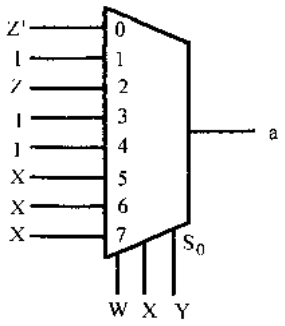


图 5-69

5.34　多路转换器实现的花费 = 7($0.25 + $0.35) = $4.20; ROM 实现的花费 = $1.25 + $0.35 = $1.60。

5.35　$S_1S_2 = 01, S_3S_4 = 00, S_5S_6 = 11, S_7S_8 = 01$

5.36　点 m 连在点 F 上。

5.37

门类型	数字	芯片内容	所需芯片数
3-输入	36	3 门	12
4-输入	4	2 门	2
反相器	18	6 门	3
			总计 17
多路转换器	4	2	2
分用器	5	2	3
			总计 5

5.38　7 个门芯片花费 $3.40, 并且安装花费为 $5.95; 总花费为 $9.35。需要的面积是 17 in^2。五个多路转换器和多路分配器芯片花费为 $4.00 并且安装花费为 $1.75。总花费为 $5.75。需要的面积仅为 5 in^2。花费持平点是多路转换器的花费为 $1.52(忽略面积优势)。

5.39　由第 4.27 题可知, 需要 5 个芯片; 因此, 用单个门电路实现的花费为 $2.75。这使得 ROM 的价钱高达 $1.90。

5.40　$2^8 \times 8 = 2048$。大约有 2000 种可能的连接, 所以该器件称作 2K 的 ROM。

第6章 时序图

6.1 引言

通常利用时序图来描述和研究逻辑系统的特性。这些逻辑图沿着时间轴显示数字电路不同点上的电压或逻辑电平，并且可用来表示存在于输入和输出间的函数关系和时间延迟。输入变化引起相应输出变化的速度经常是非常重要的设计参数，在研究它对系统性能影响方面时序图是有用的帮助。在逻辑硬件冲突中逻辑分析仪是广泛使用的实验仪器，它是同时显示多个时序波形的多通道存储示波器。特别在设计阶段另一个有价值的工具是能在计算机工作站上运行的软件组成的逻辑仿真仪，它能从原理图或布尔输入中产生一组相关的时序图。逻辑仿真仪对设计者而言是有价值的，因为它在诉诸于硬件前先验证系统的性能。

有两种基本的时序图类型。第1种是关于在单个门电路中遇到的传输延迟的微时序图。用它来检测和显示不希望的条件，例如，假信号脉冲和由于反馈引起的不稳定振动。第2种是关于系统中不同点处信号间关系的宏时序图，它在足够大的时间尺度上没有必要考虑单个门电路的延时。不同门的输入输出波形以固定时间间隔显示，通常以时序或时钟波形为参考。

6.2 微时序图

在数字系统中，逻辑门的硬件实现包括在导通和不导通之间切换的电子器件。因为电子放电存储和导通会影响在门输入端施加电平变化和在门输出端可识别的响应之间引起可测量到的延时，因此发生这样的切换需要很小但有限的时间。称作传输延迟时间(t_{pd})的这段时间间隔由硬件生产者在其数据手册中具体说明，并有望成为适合于所有相同类型门的常数。

对于微时序逻辑图分析的目的，经常假设系统中每个门有相同的单位传输延时。微时序图考虑到这些传输延时，按照一个或多个输入的指定变化来表明系统的特征，并假设在 $t=0$ 时开始发生。

微时序图准备

1. 一定要知道加在系统所有输入的信号初始值(逻辑或电压)并记录为 $t=(0-)$，它是在 $t=0$ 时输入即刻变化之前的时间。

2. 用这些值，就能够确定每个内部门的状态，并将其记入时序图中。

3. 允许指定的输入信号改变逻辑状态，将这个改变记入时序图 $t=0$ 处。

4. 通过研究逻辑图可以识别出将变化了的信号连到内部门的所有输入上。

5. 考察每一个门，看它的逻辑状态是否被第4步的变化改变了。如果是，将变化记在时序图中下一个连续传输延迟间隔上。提示：因为假设所有门都具有相同的延时，所以在等于单位传输延时时间 t_{pd} 的离散步长上构建时序图。

6. 如果有的话，现在将每一个改变的门输出看作是它所连接的任何门的输入变化。

7. 当发现不再有门输出变化时，过程终止。

例6.1 假设图6-1所示的逻辑电路中的物理器件(门或反相器)具有相同的传输延时。例中所有输入和输出都认为是HT，而且最初A、B、C和D都是1。当 $t=0$ 时，C变为逻辑0。画出正确的逻辑微时序图，然后将它转化来显示电压波形。

如果在逻辑1状态下涂上阴影，将不同波形(特别是手画时)分开就会在一定程度上减少混淆，见图6-2。

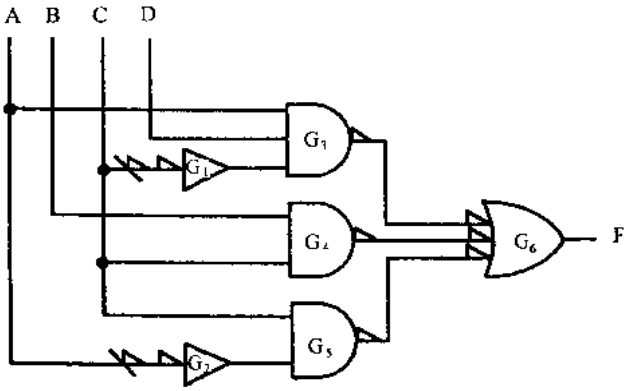


图 6-1

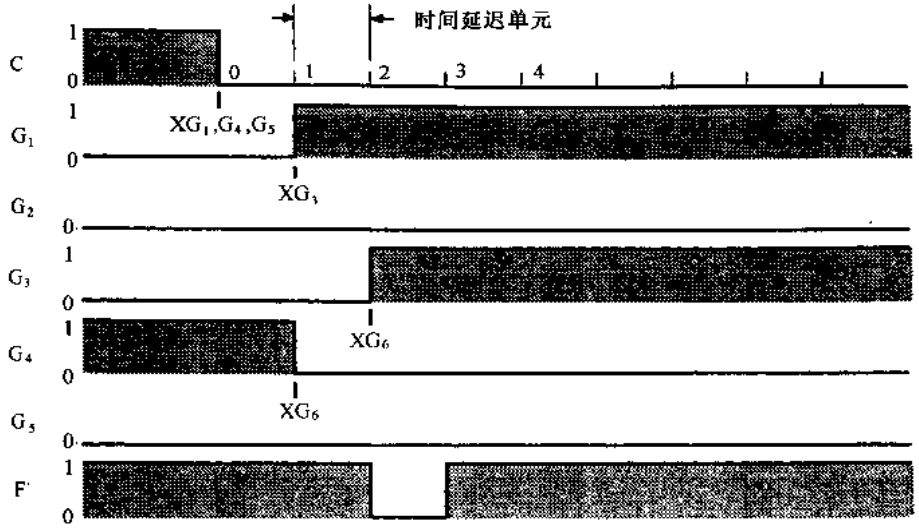


图 6-2　逻辑波形图

逻辑波形

可以看到在 $t=0$ 时由逻辑 1 变为逻辑 0 的变量 C 连在 G_1, G_4 和 G_5 门上，并且在图 6-2 上由记号“XG_1, G_4, G_5”来标识。轮流考察它们中的每一个。门 G_1 是电压反相器。因它和斜划线连在一起，所以当 C 变为 0 时它的输出一定要变成逻辑 1。提示：如果仅仅用反相器来和半箭头匹配(如没有用相联系的下划线表明)，那么就不会出现逻辑反相器而且硬件反相器在 1 个时间间隔的延时后在它的输入端仅仅传送逻辑变化。G_4 是与门，因此任何输入变为 0，它的输出将是逻辑 0。因此 G_4 会变化。G_5 因为它的输入之一已经是 0，所以不会受 C 变化的影响。将 G_1 和 G_4 输出端的变化记在图中标记为 1 的时间上，它是跟在引起它们仿真的(C 上的变化)一个时间延时间隔之后的。

现在我们看一下刚刚变化的门。G_1 门仅被连在 G_3 上；可看到 G_4 仅影响 G_6。因为 G_3 门所有 3 个输入在紧接着时间间隔 1 都将为 1，所以 G_3 门也将为 1。因为 G_6 门所有 3 个输入在紧接着 1 个时间间隔之后都将为 0，所以 G_6 门也将为 0。将 G_3 和 G_6 的变化记在时间标记 2 上。

重复以上过程，注意到仅仅 G_3 和 G_6 间的连接保持到被估计：因为所有 3 个输入会再一次是 1，所以预测 G_6 门(F)将返回到逻辑 1。因为要改变的最后 1 个门没有被连到其他任何门上，过程终止。

电压波形

G_1, G_2 和 G_6 的输出是 HT, 因此它们的逻辑和电压波形将是相同的。G_3, G_4 和 G_5 具有 LT 输出, 这使得将它们的波形取反是必要的, 从而得到图 6-3 所示的相应的电压波形。

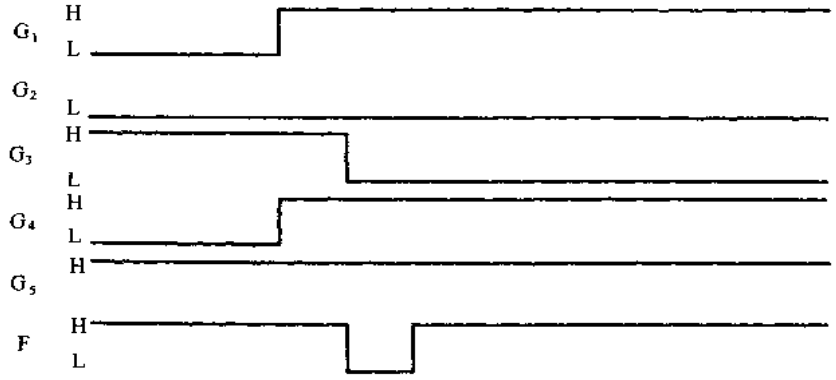


图 6-3 电压波形图

6.3 冲突

可将图 6-1 代表的逻辑很容易确定为

$$F = AC'D + BC + A'C。$$

当初始时, A = B = C = D = 1, 这个布尔表达式预测了 F 将是 1 并且即使 C 变为 0 它将保持不变。然而波形(图 6-2)却表明一个短暂的条件, 即在该处输出 F 瞬间变为低, 然后返回由逻辑所要求的正确的高状态。由于在并行逻辑路径上不相等的传输延时而产生的短时间偏离称为竞争(有时非正式地称之为假脉冲)。

因为变量 C′不能用作输入, 这在逻辑上需要添加硬件电压反相器, 在它的信号途径上又添加了延时单元。这样就产生了图 6-2 所示的冲突, 这是由于会聚在 G_6 处的 C 和 C′路径间的延时不同而产生的。

在卡诺图上可将这种竞争类型识别出来。考虑例 6.1 函数的卡诺图(见图 6-4)。

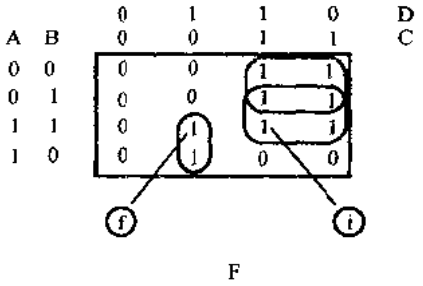


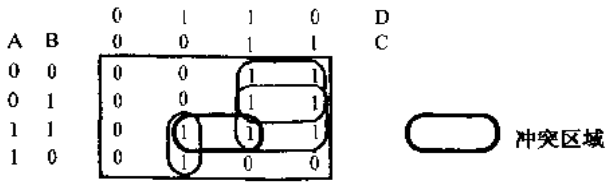
图 6-4

图中, 从最初状态 ① 到最终状态 ⑥ 的转变是穿越两个不重叠区域的边界发生的。通常情况下, 为了确定任何单独变量的变化是否是引起冲突的条件, 仅需要在卡诺图上寻找邻近的初始和最终状态, 它们在各自的区域中是分开的。

这种情况可用相对容易的方法修改过来。如图 6-5 所示添加了与源和目的重叠的冲突包围圈。这导致了在 F 的方程中添加了一项, 硬件需求上也有相应的增加。

提示: 用卡诺图来检测在组合逻辑中由于变量和它的互补变量的路径长度有差异而引起的竞争是很重要的, 也有其他的不易检测的时序问题类型(参考第 6.6 和 6.7 题)。进一步说, 虽然卡诺图可以表明在冲突中涉及到的两个输入状态, 但它不能识别哪一个是初始状态。很有可能从状态 S_1 到状态 S_2 时会发生冲突, 但另一个方向却不会发生, 如第 6.1 题所示。

冲突常常是不可避免的, 除非有正确的设计, 冲突会在数字电路中引起严重错误。处理它们的有效工具涉及到第 7.4 节讨论的脉冲和时钟技术的实现。



$F = AC'D + BC + A'C + ABD$

图 6-5

6.4 宏时序图

宏时序图用来显示相对独立的输入或其他参数如：系统时钟脉冲对不同门的输出，在研究传输延时现象时主要关心的是比它大得多的时间尺度上的电路特性。单个门延时的影响经常被特意忽略掉，或者与如图所示的相关逻辑连续转换相比变得非常的小。

例 6.2　连到图 6-6 所示逻辑的 HT 输入 A, B, C 和 D 随时间的变化如图 6-7 所示。假设电路对输入变化同时响应(也就是，在所使用的时间尺度上不能辨认出器件的传输延时)，画出 F 和 G 的输出电压波形。

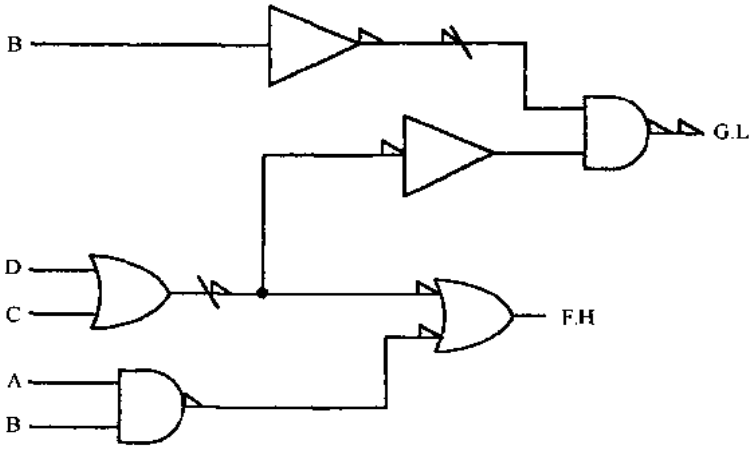


图 6-6　逻辑图

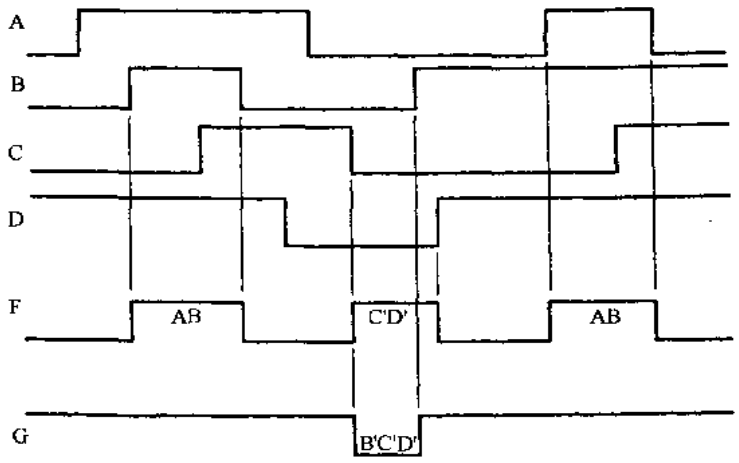


图 6-7　电压波形图

从逻辑图中可知，F = AB + (C + D)′。在第 2 项上利用摩根定律产生了 F = AB + C′D′；也就是当 A 和 B 都为真或 C 和 D 都为假时 F 将为真。对于给定的输入有 3 次满足了这些条件，而且因为 F 为 HT，它相应的电压波形符合逻辑波形，如图 6-7 所示。

可以看出输出 G 和 B′(C + D)′相等，或者等价地，G = B′C′D′ 。因此只有当 B，C 和 D 都为假时 G 为真，而且因为 G 为 LT，它的电压波形将是由逻辑表示的波形的反。

6.5　时序仿真

证明逻辑设计功能的主要工具是计算机逻辑仿真器，它使得设计者能在模拟设计试验和测试阶段不需要硬件。虽然不是实际样机的替代，但仿真器可以通过基本错误和时序问题的早期检测来加速设计过程。坐在工作站监视器前，工程师利用称作原理图输入的过程来组装硬件电路，通过从软件库中选择逻辑器件，用鼠标或相似的功能在计算机上显示，用对实物方便定位的指向工具将器件连起来，这样在显示器上就画出了电路图。典型的屏幕显示来自于称作 LogicWorks™的仿真包，它来自于 Capilano Computer，如图 6-8 所示，它描述了例 6.1 的电路。

在图 6-8 所示的屏幕上，表示出了提供进入门电路库的下拉菜单。在此情况下，已经选中了一些符号，强调分析将是硬件仿真。虽然目前大多数的仿真器使用正向逻辑规则，但越来越多的仿真器开始将混合逻辑符号包含进去，希望 LogicWorks™的未来版本也将这样做。

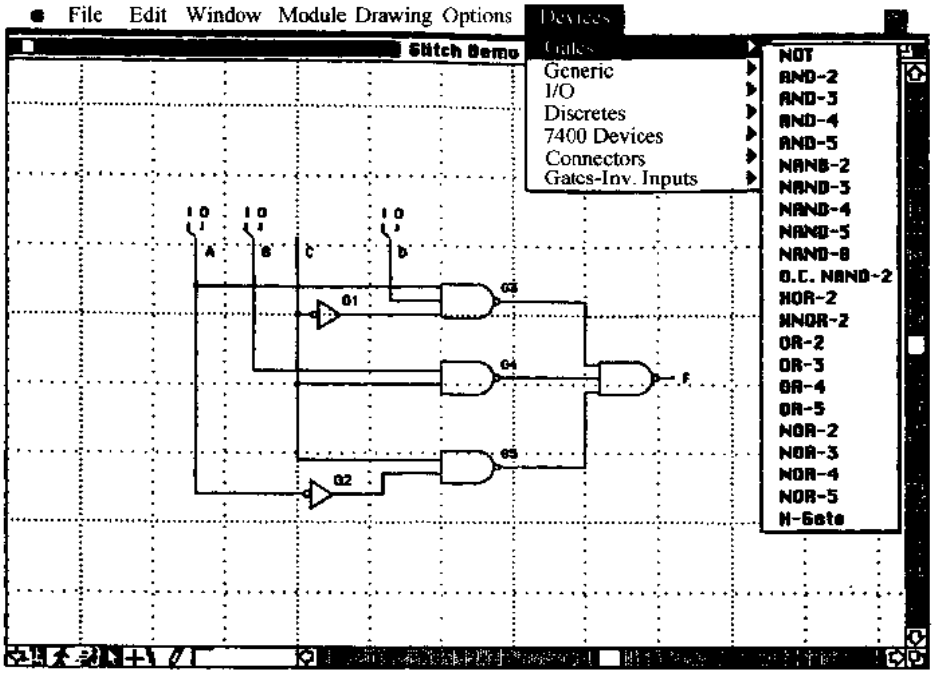


图 6-8

当设计者创建电路图时，计算机中将互连路径存为网络列表。仿真软件在网络输入端使用所选的信号和/或固定电平，并且在输出端和其他所要求的任何内部点上监测响应。图 6-8 中，输入 A，B 和 D 借助于从 I/O 菜单中选择的可操作开关连到逻辑 1 上。通过时序图窗口画出输入 C 的波形(用鼠标)使它从 1 跳变到 0，一部分波形在图 6-9 中所示。接着是停止命令，启动仿真，在不到 1 秒钟的时间内产生了所示的波形图。将该结果与图 6-3 人工画的时序图相比较，可知证实了在输出端 F 预测到的冲突。

在大多数仿真软件中，使所选的元器件有不同的延时并进行冲突捕获是有可能的。本例中，在运行仿真前将所有门的延时设为 10 个时间单元。很多软件都包括能预测冲突条件的冲突检测规程。

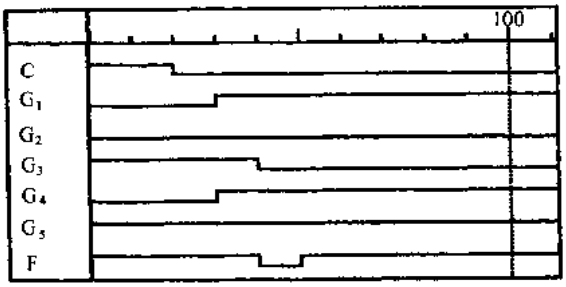


图 6-9

6.6 组合电路中的反馈

图 6-10 给出了含有反馈的组合逻辑。在这种情况下，两个输出变量(Y_1 和 Y_2)是包括它本身的几个变量的函数。换句话说，在带有反馈的电路中，给定变量的值受变量本身的影响。

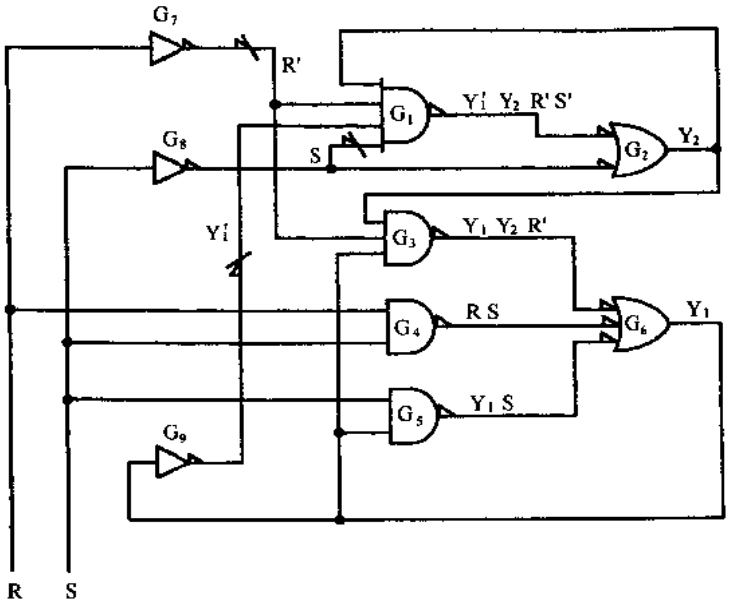


图 6-10

在某些条件下，因为反馈信号的到来会引起输出变化，会依次改变输入从而再一次引起输出的变化，如此往复，所以这样的电路可能不稳定。如果卡诺图冲突分析预测了冲突，我们可以预测在两种状态之间连续振荡的可能性，振荡规律取决于电路的传输延时。

从图 6-10 中可以看出

$$Y_2 = Y_1'Y_2R'S' + S$$

$$Y_1 = Y_1Y_2R' + RS + Y_1S$$

Y_1 和 Y_2 的卡诺图如图 6-11 所示，这里我们发现，在 Y_2 的卡诺图中，没有包含邻近 1 的重叠区域。这表明当 R = 0 和 S 变化时，在 Y_2 中会预测到冲突，建议应进一步分析电路来检查振荡特性。

很明显，当电路设计变得越来越复杂并包含反馈时，微电路图的手工构建变成一项繁重的工作。计算机仿真器使工作极大地简化，允许设计者在投资硬件前更加彻底地检查电路。例如，考虑用与非硬件来实现图 6-10 的反馈电路的情况。如图 6-12 所示，用 LogicWorks™仿真

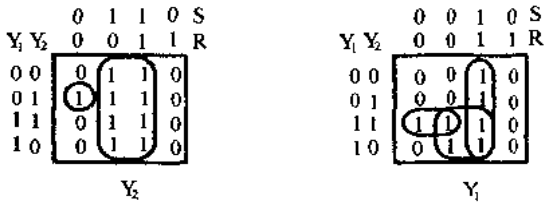
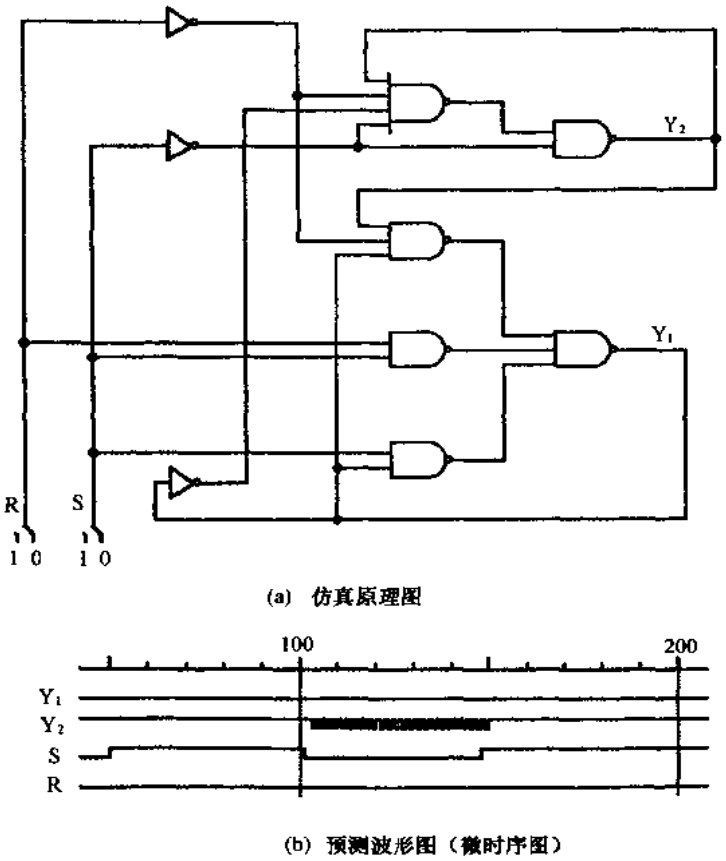


图 6-11



(a) 仿真原理图

(b) 预测波形图（微时序图）

图 6-12

可以很容易地说明预测到的不稳定性*。

提示：同卡诺图预测的一样，当 R＝0 和 S 变化时，不稳定性在 Y_2 上发生了。

习 题 解 答

6.1　考虑函数 F＝(AB)′＋AD 的 NAND 实现，如图 6-13 所示。假设所有的门具有相同的时间延时，画出 A 发生变化而 B＝D＝1 情况的微时序图。

解　开始一步步地进行信号跟踪前，建议确定其内部逻辑，利用第 4.10 题讨论的转化过程可以很容易的做到，这给我们提供了所需要的相应的混合逻辑图，在图 6-14 中重新画出。

在例 6.1 中说明了微时序图的产生。跟踪每个变量变化辐射的连接，识别出相关的门或多个门，

* 具有反馈的电路的模拟有时会失灵是由于其缺乏不明确的初始条件。这个问题的简短论述会在附录 C 运用的 Logicworks™时处理。

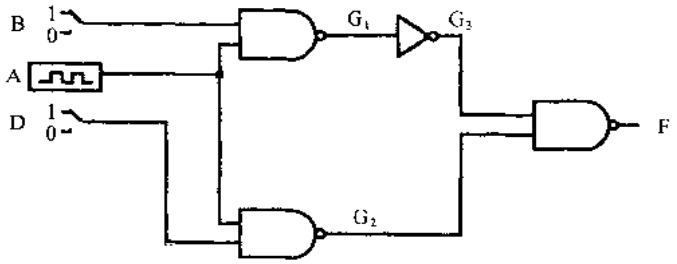


图 6-13

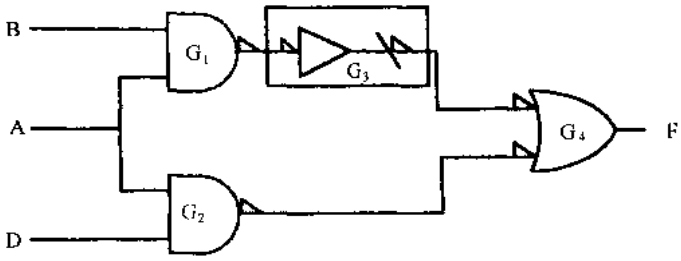


图 6-14

将逻辑电平的变化(如果有)按照仿真的每 1 个时间间隔记在图中。结果时序图如图 6-15 所示。已知的布尔表达式使我们相信不管 A 的状态如何，F 将保持等于 1，而且正如所预测的，对输入 A 从 0 到 1 的变化没有反应。然而，与布尔预测相反，在 A 处 F 1 到 0 的变化时，在随着输入的变化两个时间延时产生了冲突。如果将所有的 LT 输出反相，可将这个结果与仿真波形比较。

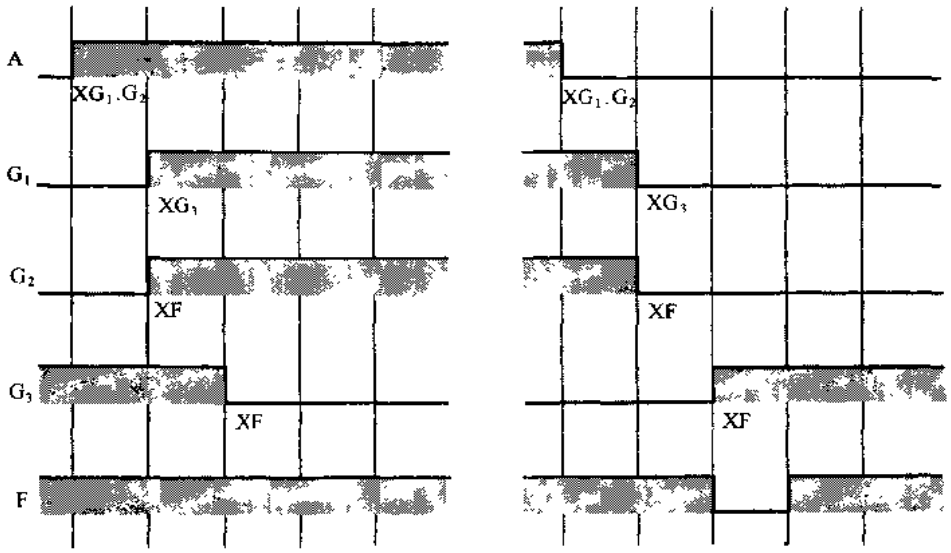


图 6-15　微时序逻辑电平

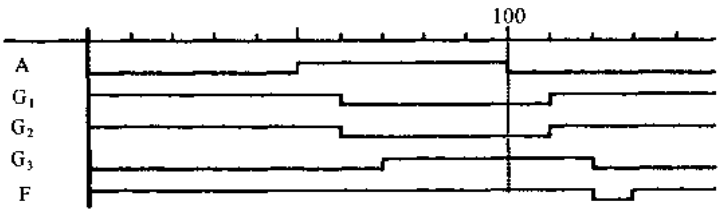


图 6-16　时序仿真。提示:A 从 1 到 0 的转变之后的 F 处的冲突

6.2　画出第 6.1 题电路的卡诺图,添加冲突包围圈来去掉冲突,并且讨论它对硬件的影响。

解　对已知函数运用摩根定律产生了 F = A′ + B′ + AD。相应于这个布尔形式,带有包围圈的卡诺图如图 6-17 所示。这里正如所标出的相邻的 1 处在分开的包围圈中,所以可以很容易识别冲突。可以在这两个标出的 1 上加冲突包围圈,或可将它扩大包围整个右边的列。前者需要添加 1 个与门。后者产生了简化的布尔项,但它需要 1 个反相器来平衡半箭头,因此,假定一些其他的说明和设计原则不推翻这种平衡,本例中可能没有多大理由选择另一种方法。

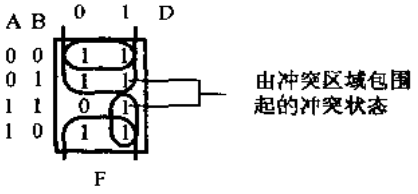


图 6-17

6.3　假设图 6-18 中所有的硬件(门和反相器)具有同样的时间延时,画出完整的逻辑微时序图。初始时,所有输入都是 1。在 t = 0 时,C 变为 1,所有输入输出都是 HT。

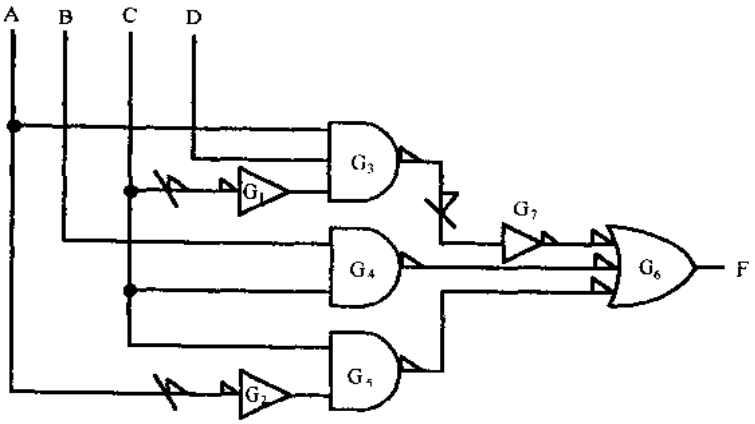


图 6-18

解　因为 A,B 和 D 保持为 1,跟踪逻辑是相对容易的。当 C 变化时,它直接影响门 G_1,G_4 和 G_5,如果它们响应,则会在 1 个时间延时间隔之后。门 G_5 因为它的低输入端保持为逻辑 0,所以它不改变。当 G_4 改变时,它不会影响 G_6 因为 G_7 处于逻辑 1,然而,反相器 G_7 响应 G_3 门的变化在第 3 个时间间隔时变为 0,这引起了 G_6(F)在第 4 个时间间隔时变为 0。逻辑微时序图如图 6-19 所示。

将这个电路与图 6-1 中的仅缺少反相器 G_7 的电路相比较是很有趣的。在图 6-18 中,虽然变量 C 的汇合路径在上面电路的分支有 3 个时间延时,而下面的电路分支仅有 1 个时间延时,因为逻辑结构的关系而不会发生冲突。

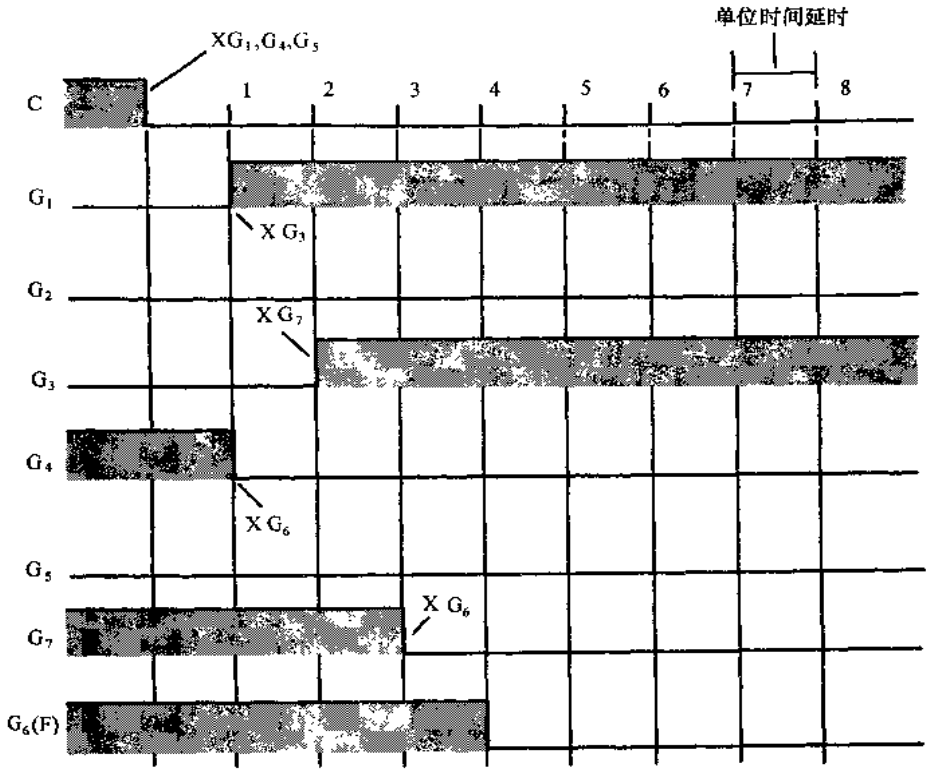


图 6-19　逻辑波形图

6.4　图 6-20 所示电路称作锁存器是通用门电路的互连，将在第 7 章中详细讨论。提示：交叉连接的反馈安排中，1 个门的输出用作另 1 个门的输入。假设输出状态如图所示，并且两个输入都初始化为逻辑 0。在 t = 0 时，输入 R 变为 1。假设两个门有同样的延时时间，画出微时序逻辑图。

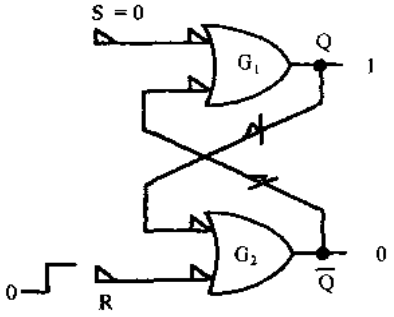


图 6-20

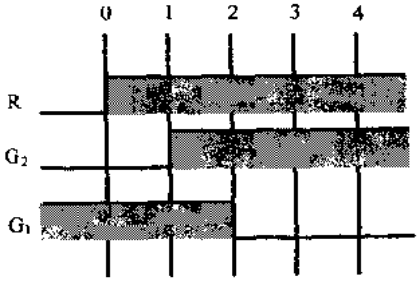


图 6-21　锁存器的逻辑波形图

解　当两个外部输入是逻辑 0 时，因为逻辑取反，G_1 和 G_2 门的反馈输入分别是 1 和 0，可看到给定的输出为恒定的。当 R 改变时，它影响 G_2 使其在 1 个时间延时间隔之后变为逻辑 1。这使 G_1 的反馈输入变为 0，并且此时门的两个输入端为 0，它的输出 Q 会在又 1 个延时间隔之后由 1 变为 0（参考图 6-21）。

我们看到 G_2 门不会响应 G_1 的变化，因为它已经处于 1 的状态，过程终止。然而借助于来自 G_1 取反的新的 1 的到达会将 G_2“锁存”在逻辑 1 的状态，因此即使 R 返回到逻辑 0，它仍然保持不变。关于它的更多内容将在第 7 章中讨论。提示：电路稳定到最终状态需要两个时间延时间隔。

6.5　图 6-22 中，初始条件是 Q = 0，A = 0，B = 1 和 C = 0。在 t = 0 时，输入 C 变为 1，6 个时间延时间隔之后，它变为 0。画出 12 个时间延时的微时序图，假设所有门延时是相同的。

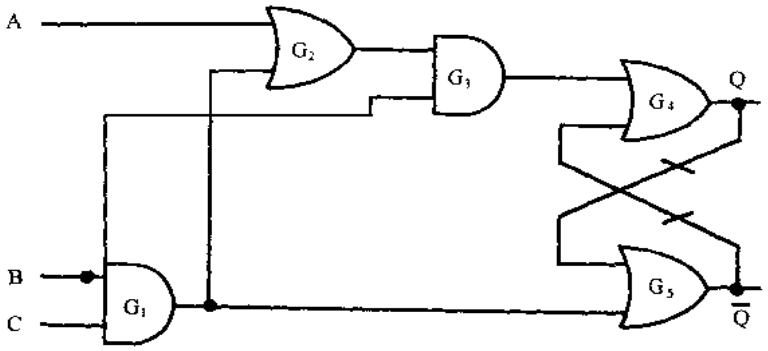


图 6-22

解 ☞ 根据假设的终止条件，利用简化的布尔关系可以推导出电路中所有门的逻辑状态。将这些状态记入 t=0 之前的时序图中，此时使输入 C 发生跳变。注意到每一步变化可能影响的那些门之后，画出 1 个时间延时之后的正确的响应，如图 6-23 所示。

在时间刻度为 8 处 G_5 的变化没有影响，因为 G_4 借助于它来自 G_3 门的输入已处于 1 的状态。因此过程终止，直到另一个输入变化。

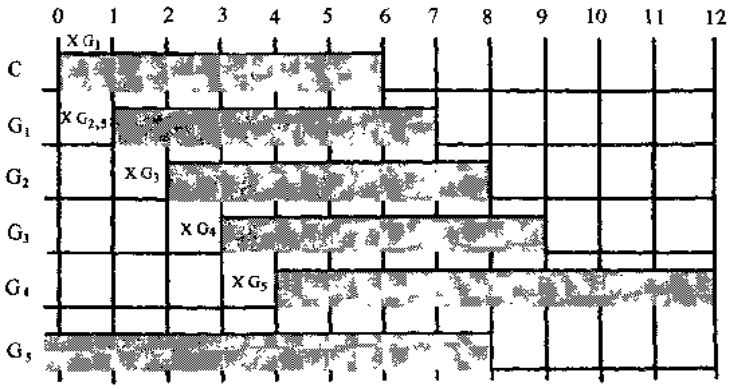


图 6-23

6.6 如果将硬件反相器插入到锁存器的一条反馈路径上，如图 6-24 所示，说明电路会变得不稳定发生连续振荡。

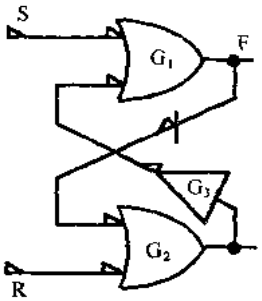


图 6-24

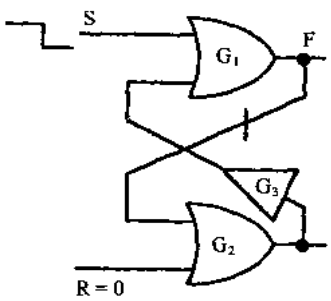


图 6-25

解 ☞ 如果我们不考虑 HT 和 LT 指定，可以很容易地展开分析。因此，我们会首先通过去掉半箭头将图 6-24 的硬件图转化成逻辑图(见图 6-25)。通常也将反相器去掉，因为它们仅仅用作将 LT 转

化为 HT 或相反，而不进行逻辑上的转换。然而本例中，特意研究传输延时的影响，必须考虑硬件反相器 G_3 的延时，虽然我们在逻辑上认为它是一条导线。

从图 6-25 中应该清楚的是，在锁存电路中已引入了逻辑上的不对称。输出 G_1 的变化在传输到 G_2 前已被逻辑取反，而 G_2 输出的变化传输到 G_1 输入端。如果 S = 1，G_1 的输出将是 1，G_2 的输出将是 0，R 上的变化没有影响，因为或门 G_1 仅需要一个为真的输入使它的输出为真。电路是稳定的。另一方面，如果当 R = 0 时 S 由 1 变为 0，如图所示，那么我们看到 G_1 门将有两个逻辑 0 的输入，在 1 个时间延时之后 F 将为 0。认为在 G_3 的输入端(注意斜划线)为 1，在又 1 个延时之后它的输出变为 1，这个 1 通过 G_3 传输，G_3 又增加了 1 个传输延时。因此在 S 发生变化 3 个传输延时之后，G_1 接收到取消信号返回到 1 状态。当然这个变化又一次启动了连锁反应，认为出现了重复周期为 6 个传输延时的振荡。正确的微时序图如图 6-26 所示。

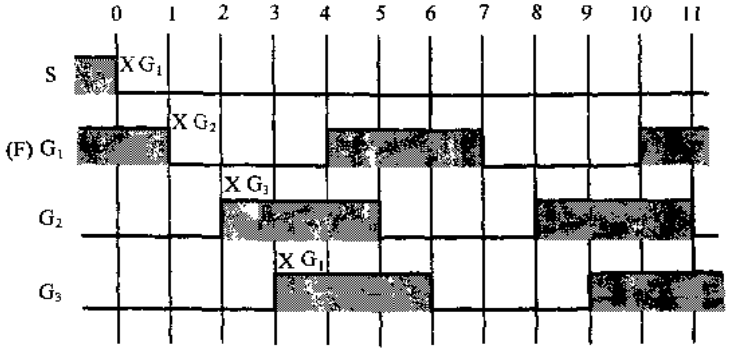


图 6-26

因为微时序图以每次一步画出，这使得进行跟踪和综合的连续分析比仅用文字试着推理的过程(它们可能表达地很好)更加容易。

6.7　说明如果输入变化的持续时间减少至 1 个时间延时，也会导致图 6-20 的锁存器电路不稳定。

解　同第 6.4 题一样，假设当 Q = 1，Q′ = 0 时输入 R 由 0 变为 1。紧接着，如果它触发的信号传输通过 G_1 并且以状态 1 回到 G_2 的输入端，锁存它之前，R 就返回为 0，那么 G_2 就返回到 0 状态而不是保持在状态 1 上，进一步分析表明，会发生连续振荡直到 R 或 S 变为逻辑 1。逻辑波形如图 6-27 所示，其中箭头表明了涉及到的因果关系。这个问题说明了这样的事实，就是设计者为了保证电路的稳定性必须考虑输入的持续时间和连接路径。

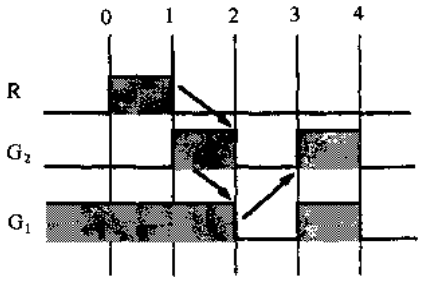


图 6-27

6.8　画出图 6-28 所示电路的宏时序图，逻辑波形如图所示。

解　$C = D \oplus S_0$ 以及 $H = (XS_1)'$。当 H = 0 时多路转换器的输出是 $F = XS_1$，当 H = 1 时 $F = GH$。这些关系在时间上与给定输入变量相关，如图 6-29 所示。

6.9　如图 6-30 所示的逻辑电路代表主-从 JK 触发器(将在第 7 章中详细说明)。其外部引脚(用小方框来表示)的输入状态如下：J = 1，K = 1，CLK = 0，CLR = 0，PR = 0，Q = 1，Q′ = 0。画出微时序图，说明在 t = 0 时 CLR 输入由 0 跳变到 1 时将发生什么。

解　在开始进行时序图之前，有必要确定每个门在 t = 0 时的状态。从给定的输入条件推理如下：

- 因为 Q′ = 0，G_8 的所有 3 个输入一定是 0。那么 $G_6 = 0$。
- 因为 G_7 门的低输入支路为 1，所以 G_5 门的状态仍然不能确定，这足以说明给定的条件 Q = 1。
- 因为 G_6 的另一个输入是 1，所以 $G_{10} = 0$。

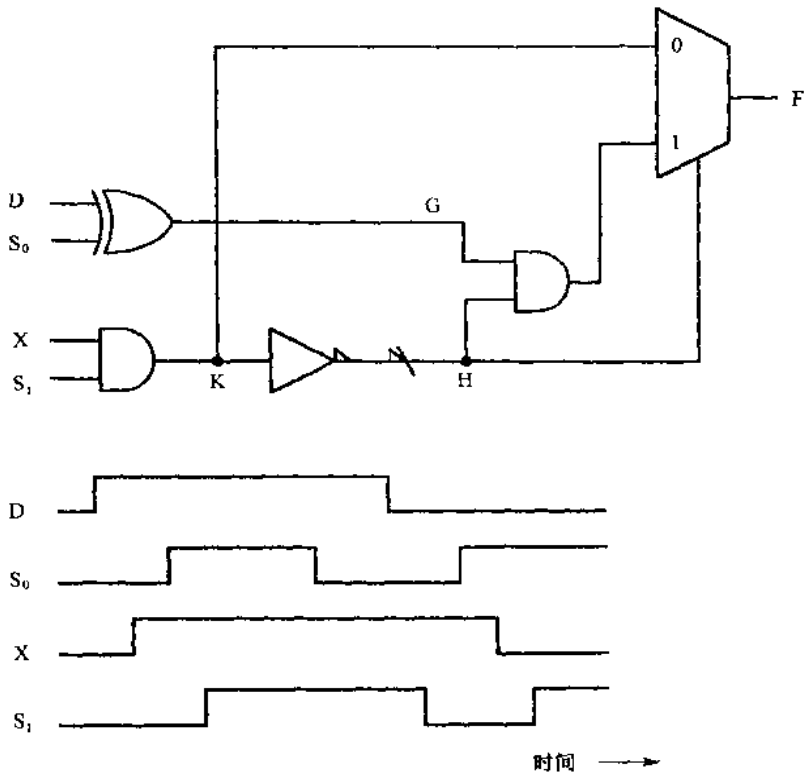


图 6-28

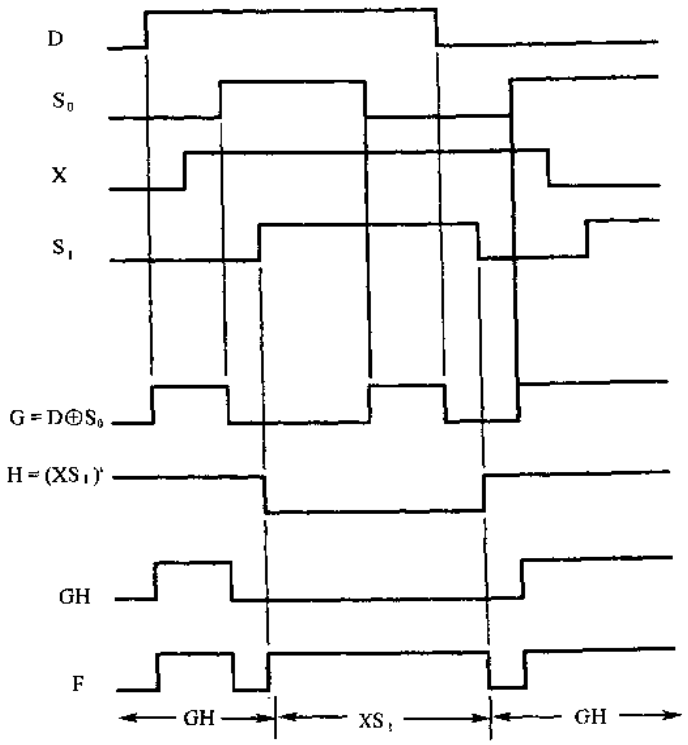


图 6-29

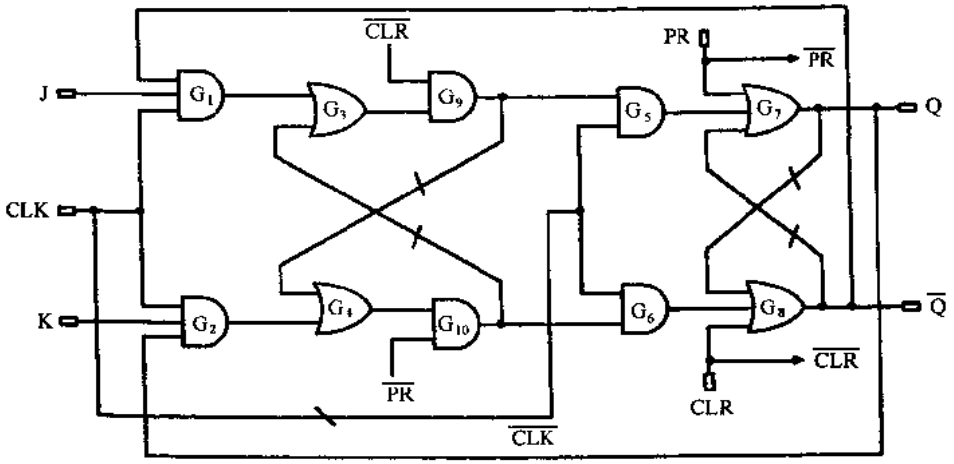


图 6-30

- 因为 G_{10}的另一个输入是 1，所以 $G_4 = 0$。
- 因为 CLK = 0，所以 $G_1 = 0, G_2 = 0$。
- 通过 G_{10}的反相器 $G_3 = 1$。
- 因为 G_9 的两个输入都是 1，所以 $G_9 = 1$。这也可以从 G_4 门的输入要求推导出来。
- 因为 G_5 的两个输入都是 1，所以 $G_5 = 1$。

在 t = 0 时，CLR 输入由 0 变为 1。一个传输延时单元之后，G_8 和 G_9 在 1 个时间间隔后响应，如图 6-31 所示。

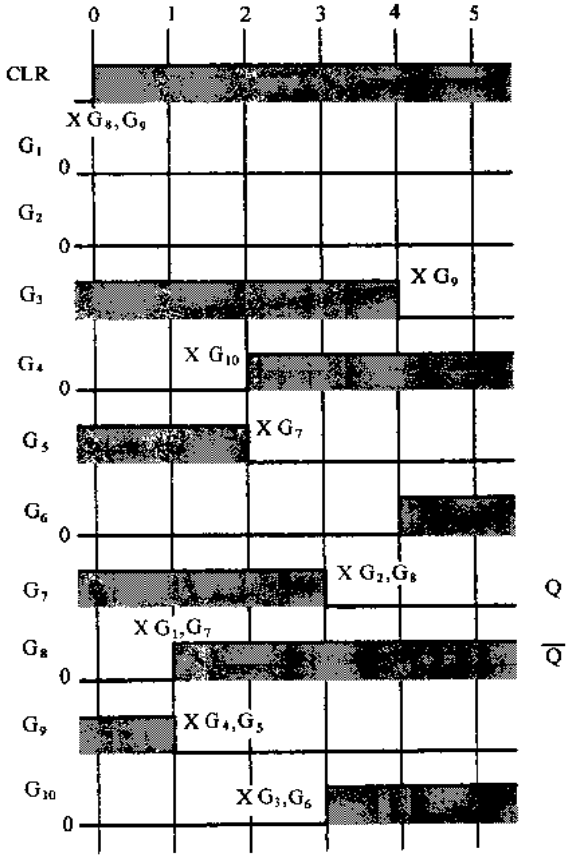


图 6-31

响应继续传播。

• G_8 被连在 G_1 和 G_7 上。因为 CLK 保持为 0 所以 G_1 不能改变；因为来自于 G_5 的输入 1 所以 G_7 不会改变。

• G_9 被连在 G_4 和 G_5 上，这两个门在时间间隔 2 上响应。

• G_4 影响 G_{10}而 G_5 影响 G_7，这两个都在时间间隔 3 上响应。

• G_7 被连在 G_5 和 G_9 上，它们都不会响应。

• G_{10}被连到 G_3 和 G_6 上，它们在第 4 时刻上响应。

• G_3 被连到 G_3 而 G_6 连在 G_8 上，这两个门都不会响应，过程终止。

注意：这个电路需要 4 个时间延时完成响应过程。施加更短持续时间的 CLR 输入会导致第 6.7 题说明的一些不尽人意之处。

6.10 考虑一个综合问题，要求对一组给定的输入响应以产生特定的输出，详细说明在图 6-32中给出，这里假设输出 F 在 1 个传输延时之后响应输入 A 和 B。



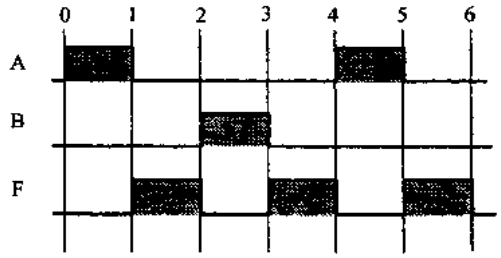


图 6-32

解 察看输入和输出间的基本关系，去掉延时重新画出宏时序图波形，如图 6-33 所示。通过读下面的时间序列关系可以得到真值表：

间隔	A	B	F
0—1	1	0	1
1—2	0	0	0
2—3	0	1	1
3—4	0	0	0
4—5	1	0	1
5—6	0	0	0

多余的

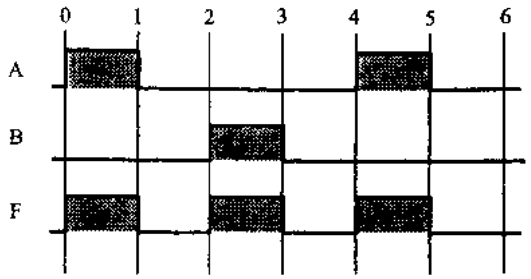


图 6-33

去掉多余的项，提示：11 状态不会发生，可将它认为是“无关项”。

A	B	F
0	0	0
0	1	1
1	0	1
1	1	X

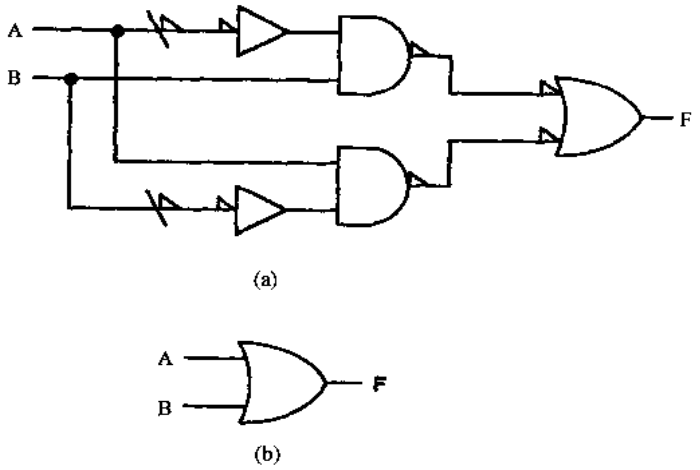


图 6-34　6.10 题的两个解决方案

综合或设计问题很少有惟一答案，设计者必须从很多可能的方法或方案中选择。例如，在这个练习中，有人会写出 F = A′B + B′A，并且用 5 个器件，如图 6-34 所示；稍微有些经验的设计者会识别出异或函数。有灵感的设计者利用 11 状态永远不会出现的知识，会仅使用 1 个或门。

应该指出综合问题通常是很复杂的，需要用比本例更复杂的方法。假设 F 在 3～4 间隔时为 1，这种情况下，一定会在 1～2 的时间间隔发生冲突，因此要运用某些其他的设计工具(见第 10 章关于状态机的内容)。

补 充 习 题

提示：第 6.11 题到第 6.13 题说明了硬件和真值指定如何影响电路时序和经济性。

6.11　图 6-35 所示的电路是在第 4.36 题中得到的。对于所有门，如果最短的传输延时是 5 ns，最长的延时是 15 ns，找出电路中最长和最短路径延时。

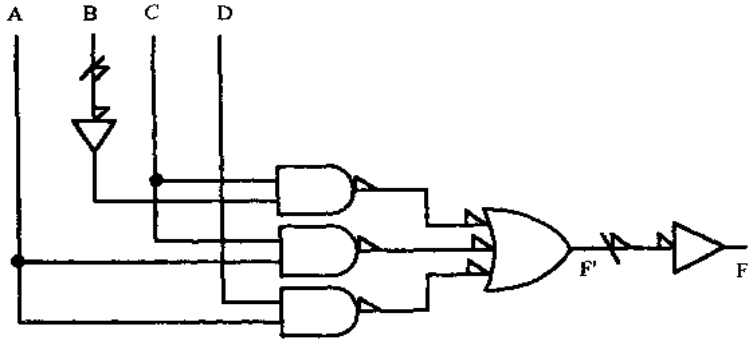


图 6-35

6.12　在第 4.38 题中得到了 F 的两个其他方案。利用在第 6.11 题中指出的相同的传输延时，找出电路中最长和最短路径延时(参考图 6-36)。

6.13　在用户芯片上制作门或反相器需要以下的资源分配：

每个门	2 个单位
每个输入，在第 1 个之后	0.5 个单位
每个互连线段	0.25 个单位

如果每个硅资源单位花费 $0.02，并且超过 15 ns 延时的路径每个单位花费 $0.10，比较第 6.12 题中两种实现的花费，假设最坏情况下门传输延时 15 ns。

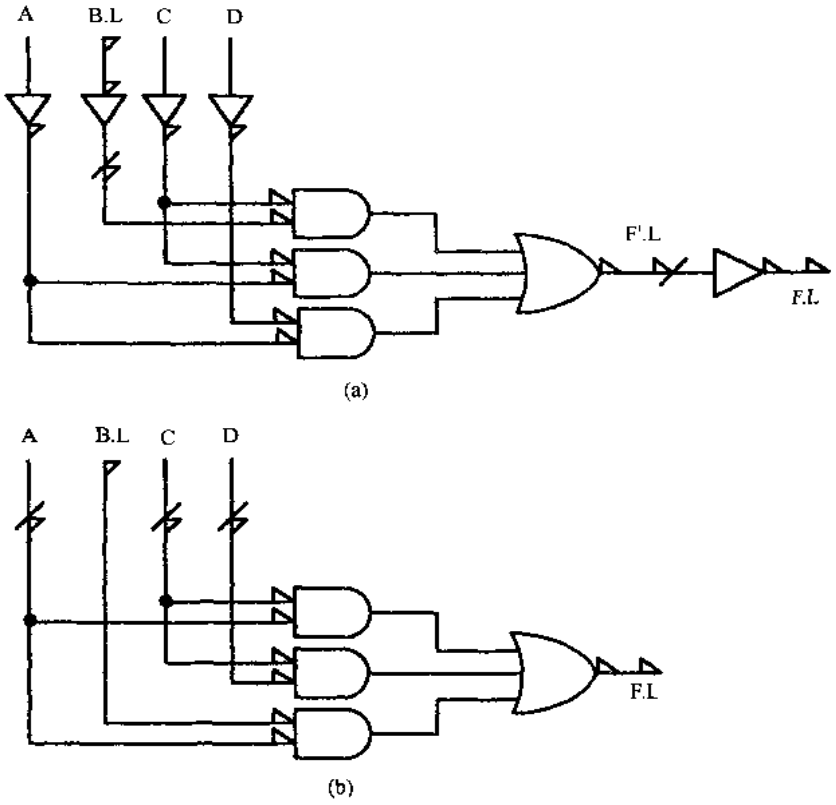


图 6-36

6.14　在图 6-37 中，所有硬件具有同样的时间延时。假设初始输入 A 和 D 为 1，并且在 t = 0 时，L 循环如图 6-38 所示，画出完整的微时序逻辑图。所有输入输出是 HT。

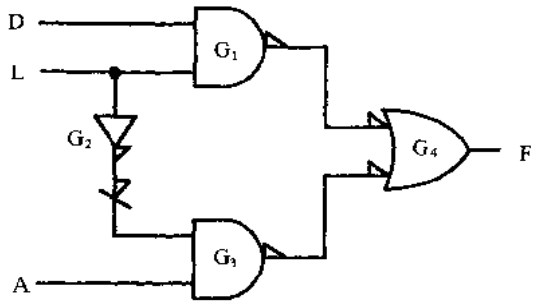


图 6-37

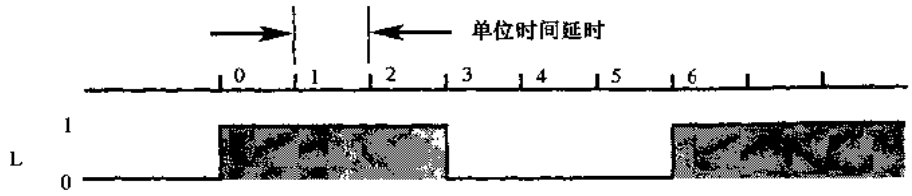


图 6-38

6.15　图 6-39 中的多路转换器在两个门-延时间隔之后响应它的选择输入的变化。假设在 t = 0 之前的瞬时，输入 R 和 S 为 0，Q = 0，多路转换器的输入 A 和 B 分别是 0 和 1。画出在 t = 0 时 S 转换为 1 时的微时序图。

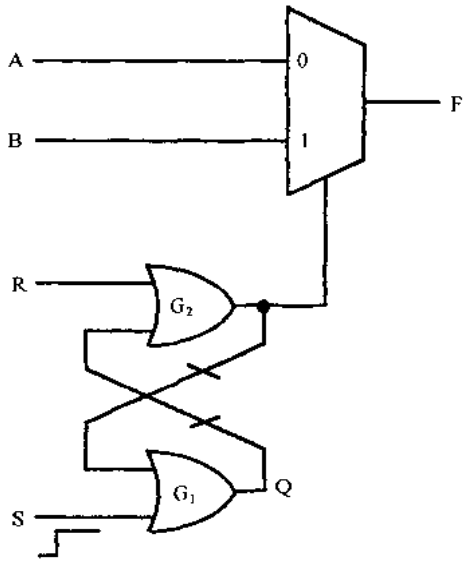


图 6-39

6.16　确定图 6-40 所示的电路是否存在任何冲突，如果有的话，确定引起冲突的转变。

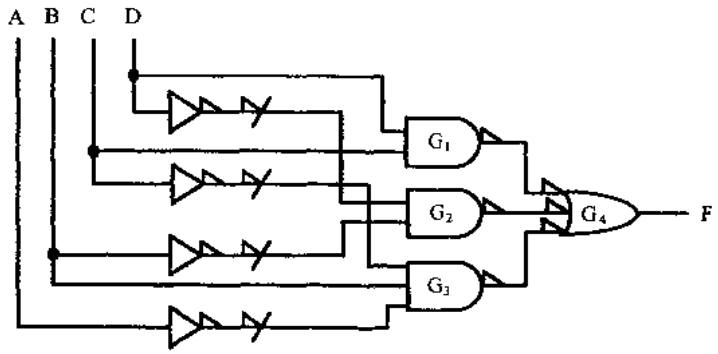


图 6-40

6.17　对于第 6.16 题的电路，画出关于 G_2 和 G_3 发生转变的电压微时序图。

6.18　参考第 6.16 题的电路，画出关于 G_1 和 G_3 发生转变的电压微时序图。

6.19　为了防止第 6.17 题的冲突发生，应对图 6-40 中的电路做怎样的改动？

6.20　在例 6.1 的电路中(图 6.1)，指出所有可能的冲突。首先观察会聚的路径，确定可能的冲突，然后用卡诺图进行彻底的分析。

6.21　如图 6-41 所示的反相器环路是一种用来检测微电路速度的电路。说明如果反相器的数目为奇数时，会产生振荡。

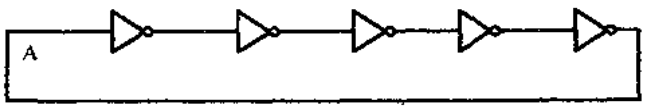


图 6-41

补充习题答案

6.11　输入变化可能花费 15～60 ns 出现在输出端。

6.12　最长和最短路径长度相等，经过网络的时间介于 10 到 30 ns 之间。提示：如何选择与真值匹配的硬件会影响电路的性能。

6.13

	(a)		(b)	
资源	24.75	$0.50	13.5	$0.27
多余时间	45 ns	$4.50	15 ns	$1.50
总计		$5.00		$1.77

6.14 见图 6-42。提示当 L 由 1 变为 0 时发生冲突，反之不发生。

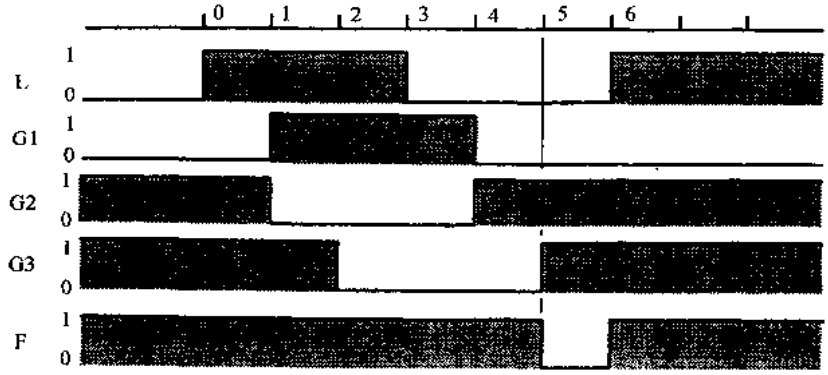


图 6-42 提示当 L 由 1 到 0 时冲突发生，反之不发生

6.15 见图 6-43。

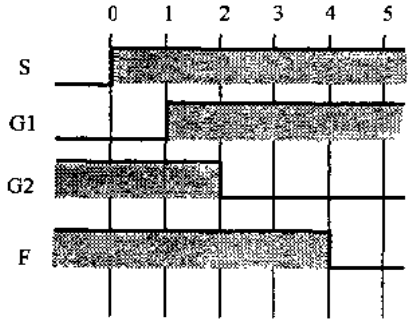


图 6-43

6.16 画出逻辑 F = B′D′ + CDA′BC′的卡诺图，表明除了非重叠区域有 3 种情况，下面的转变会发生冲突。

A	B	C	D	到	A	B	C	D
0	1	0	0		0	0	0	0
X	0	1	1		X	0	1	0
0	1	1	1		0	1	0	1

6.17 见图 6-44。找出 5 个时间单元来说明传输延时以强调冲突。

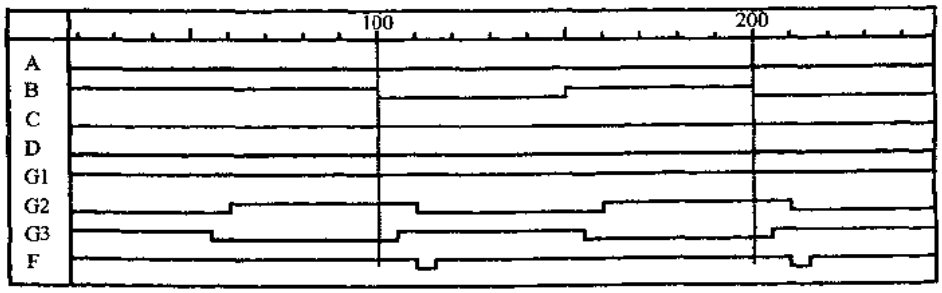


图 6-44 注意在正确的转变时的冲突

6.18　见图 6-45。所有门传输延时为 1 个时间单元。

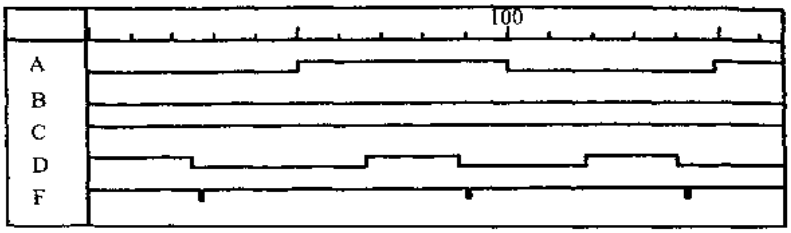


图 6-45　注意与 A 无关的冲突

6.19　通过将 G_4 转化成一个 4 输入门并且用与非门来连接 A′C′D′项，添加冲突区域，如图 6-46 所示。

		0	1	1	0	D
A	B	0	0	1	1	C
0	0	1	0	1	1	
0	1	1	1	1	0	
1	1	0	0	1	0	
1	0	1	0	1	1	

图 6-46

6.20　有来自输入的两个会聚路径；一个来自 A，一个来自 B。电路的布尔方程式是 F = AC′D + BC + A′C，当将其画成卡诺图时，表明冲突仅存在于 C(ABCD 到 ABC′D 的转变)。包括 A 上的可能冲突。

6.21　假设点 A 是 0。如果有 n 个整数，如果 n 是奇数，1 会出现在链的终端。因此，信号 A 会在 $n(t_{pd})$ 个时间延时过去之后变为 1。1 会依次变为 0，等等。仿真如图 6-47 所示。

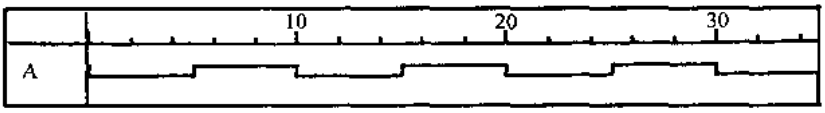


图 6-47

第7章 触 发 器

7.1 引言

除了完成组合逻辑，大多数现代数字系统也存储命令等待被处理以及作为计算结果的数据。本章讨论各种不同基本数字存储元件的实现，双稳态电路即触发器。它最原始的形式称作锁存器，由一对具有输入和输出以反馈形式互联的逻辑门组成，能存储一个单独的位。在基本锁存器上添加元器件并以不同方式组合锁存器产生几种不同类型的触发器，每一种都设计成在专门应用中具有最优的性能特点。本章包括 RS、JK、D 和 T 触发器并介绍了它们的工作方式及触发特点。

7.2 基本锁存器

图 7-1 给出了称作锁存器的简单存储器件的逻辑。

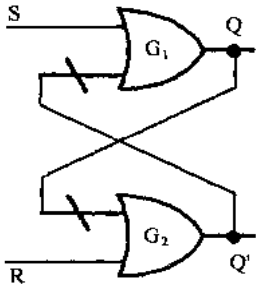


图 7-1

电路具有两个输入(R 和 S 分别对应于“置位”和“复位”)和至少 1 个输出(Q)。有时如图所示也包括第 2 个输出(Q′)。研究这个闭环是很容易的：将信号 Q 逻辑取反并连到产生 Q′的 G_2 门上。轮流地，将该信号逻辑取反并连回 G_1 门的输入端、产生 Q，因此完成了环路。当 R 和 S 是逻辑 0 时，可用这种电路形式来存储单个信息位。

当输入 R 和 S 都是 0 时，假设输出 Q 为 1。将它的逻辑取反成为 0 并反馈作为 G_2 的一个输入。现在 G_2 的两个输入都为 0，并且因此在它的输出 Q′产生 0。将这个信号取反并回送到 G_1 的输入端，因此它的输出 Q 一定是 0，从而证实了最初的假设 Q = 1。提示：在这些情况下 Q′是 0。这种情况将被保持到输入发生改变。

可用相同的分析来说明 Q = 0 也是稳定状态。在这种情况下，Q′将是 1，并且正如所期望的那样，当 R 和 S 都为 0 时，我们知道输出 Q′总是 Q 的逻辑反。

现在假设 S 变为 1，因此输入状态是 R = 0，S = 1。这样至少在 G_1 的一个输入上置 1，这使 Q 为 1。这个 1 被逻辑取反并在 G_2 的输入上变为 0，因为 R = 0，证明了 G_2 的输出为 0。将这个 0 取反并回送到 G_1 的另一个输入端，因此加强了其输出 Q = 1。在反馈过程完成它的操作并将输出保持在 Q = 1，可以去掉引起这个过程在 S 上的逻辑 1。因此，当 R = 0 时 S 上的 1 保证了 Q = 1，并且当 S 返回 0 时这个状态将会被保持。我们称电路已被锁存。用相同的方式，很容易表明 R = 1 和 S = 0 的输入产生了 Q = 0，并且当 R 返回 0 时这个状态将被保持(锁存)。

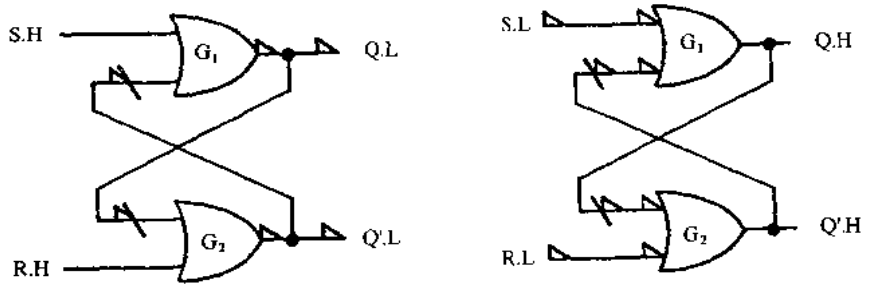


图 7-2

因此，如果将条件 S=0，R=0 认为是初态或静止状态，电路的输出指示哪个输入最后连到逻辑 1；换言之，可以将锁存行为解释为存储功能。

可用图 7-2 所示的或非或与非硬件实现简单的锁存（见题 7.17 的正向逻辑形式）。提示：在这两种情况下逻辑是相同的，而仅仅输入和输出的真值不同。

7.3　无抖动开关

数据常常通过人工操作开关的方式进入数字系统。这些开关有很多形式，从计算机键盘的单个按键下的弹簧触点到数控机床上的按钮开关。所有这些机械开关的通用特性是当动作时它们的弹性触点有抖动的趋势，从而引起持续几毫秒的短时间的接触和断开。在给定系统中，如果闭合的电气开关代表逻辑 1 而断开的触点代表逻辑 0，接触抖动会产生严重的问题，因为当想要一个单独的 1 时可能会产生多个 1 和 0。

可将锁存器作为图 7-3 所示的非常方便的“防抖动”电路。当所示的开关 SW 在上边的位置时，电压 V（相应于逻辑 1）被加到 S 的输入上。没有电流流进下面的电阻。输入 R 将对应于逻辑 0 的接地电位。因此，开关位置相应于 S=1，R=0。当开关打到下面的位置时，电压 V 被加到 R 上，输入条件是 S=0，R=1。相应地，开关抖动非常小。开关的运动部分会上下抖动。在抖动的顶端，锁存器输入是 S=R=0，在这样的条件下没有输出发生改变。因此，仅仅当手动把开关从一个位置改变到另一个位置时锁存器才会改变，并且它仅仅响应最初的接触，而忽略发生的任何抖动。

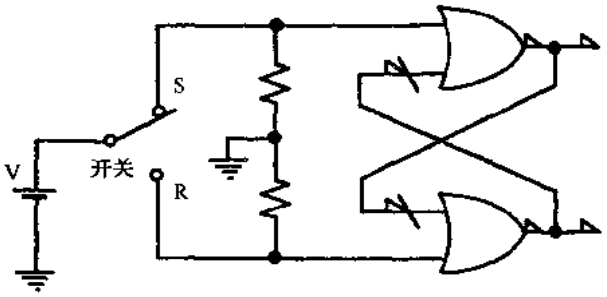


图 7-3

7.4　时钟 RS 触发器

正如第 1 章指出的，存储在数字系统中的数据易受噪声的影响。如图 7-2 所示的简单锁存器为“捕捉 1”，因为当 R 或 S 端出现 1 的时候，输入可被触发器捕捉（锁存），而不管 1 是由真正的数据引起还是由噪声引起，它会做同样的反应。

如果将 R 和 S 输入与使能信号（经常称作时钟）与，如图 7-4 所示，那么到达锁存器的输入会仅仅响应时钟脉冲持续期间 R 和/或 S 输入的噪声。许多设计者采取了允许数字线（R，S）易受噪声影响而花大力气使时钟信号相对无噪声的方法。在这种情况下，仅仅在时钟脉冲为真的时间期间，锁存器受到多余信号的影响。因此采用窄时钟脉冲会使电路噪声免疫力提高，经常将它称之为时钟控制，门控制或者脉冲选通锁存器。

经常用状态表来描述时钟 RS 触发器（表 7.1）。这里 Q_n 是时钟脉冲前的 Q 值，而将 Q_{n+1} 假设为当施加时钟电平为真时输出 Q 值。第一行表明如果 Q 和 R 都为 0，时钟后的 Q 值与前一个时钟的值相等；也就是不会发生改变。接着两行表明当 R 和 S 互补时，Q 输出与 S 相等。输入组合 R=1 和 S=1 是禁止的。这是因为可能存在输出状态不定这个条件。如果 R，S 输入都是 1 并且时钟为 1，那么两个输出都将是 1。如果问题发生了，当时钟返回到 0 后，输入变化为 R=S=0。在这种情况下，输出状态将取决于哪一个与门输出先达到 0，结果由竞争的优胜者而不是逻辑决定。

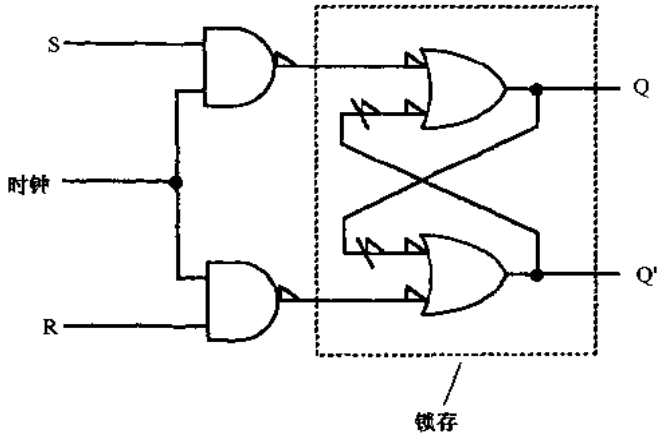


图 7-4

表 7.1

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	禁止

除了改善噪声免疫力，时钟还起到另一个有用的作用。我们可以认为它是使能输入，当它为真时，将锁存器连到 RS 数据线上。这是很重要的，因为在第 6.2 和第 6.3 节已经知道，传输延时会引起期望的逻辑电平在稍微不同的时刻出现在门输入端，甚至会引起片刻的错误信号(假脉冲波)。使能或时钟信号延迟了锁存器或其他数字电路的操作，直到它的输入有时间建立起稳定的最终值。使能信号的周期形成了数字系统的同步操作基础。使每个使能脉冲有效地一步推进了数字处理过程，并且因为这些脉冲在有规律的时间间隔发生，象时钟工作那样，术语使能和时钟的内部改变能力看来是相近的。

脉冲选通数据锁存器

如果在图 7-4 的输入 R 和 S 之间添加反相器(如图 7-5 所示)，那么参考表 7.1，表明当使能输入为真时，输出 Q 会随着输入 D 的变化而变化；并且当使能输入为假时，数据将被锁存(存储)到触发器中。

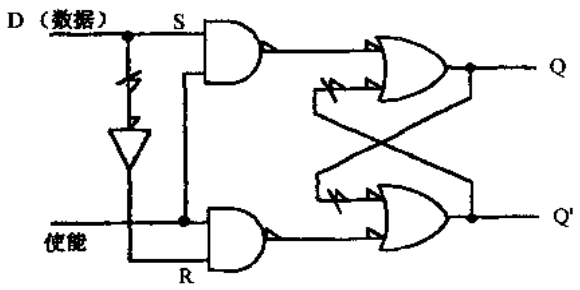


图 7-5

例 7.1 在图 7-5 所示的电路中，假设在 $t = t_0$ 时 Q 为低，当 $t = t_1$ 时施加使能选通脉冲输入，并在 $t = t_2$ 时将其去掉。给定数据波形的响应输出如图 7-6 所示，其中假设传输延时对于所选的时间尺度而言很小，可以忽略。

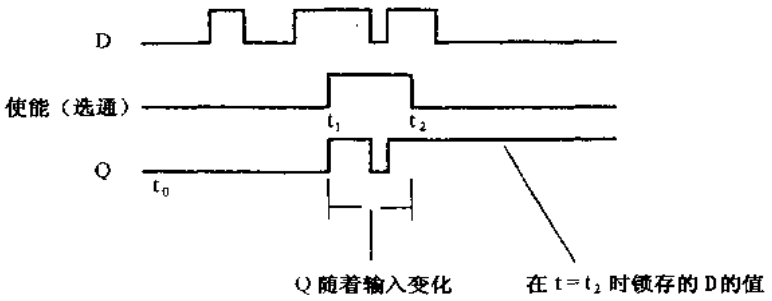


图 7-6

只要选通脉冲为 0，锁存器输入 RS = 00，没有变化发生。当使能端为 1，Q 的状态与 D 的状态相同（见表 7.1）；我们说 Q 在这段时间间隔对 D 是透明的。当使能信号在 $t = t_2$ 处变为 0 时，锁存器在 t_2 前的瞬时封锁，触发器存储 D 输入处的值。重要的是要记住所有硬件都具有传输延时并且当输入信号脉冲宽度非常小的时候一定要考虑时序因素。

每个锁存器都与多条数据总线中的一条相连；选通锁存器阵列，如果使用 n 个锁存器，并且如图 7-7 所示，由共同的使能信号同时选通，那么该阵列能够捕捉 1 个 n 位的字。

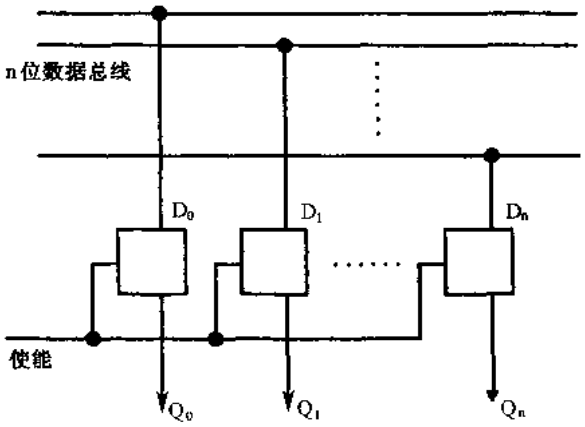


图 7-7

7.5 JK 触发器

RS 触发器的禁止状态代表状态不定。通过添加元器件，设计者已经修改了基本钟控 RS 触发器，因此，当加入时钟信号时电路会翻转（也就是对它的输出状态取反），这种修改带来了

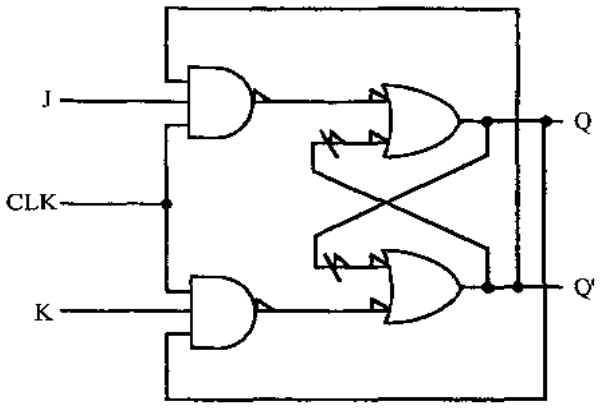


图 7-8　“错误”的门控锁存设计，输入为 J 和 K

挑战，因为必须考虑传输延时及逻辑互连。最初的尝试如图 7-8 所示。设计者将触发器的输出回送到输入门，当 11 输入条件发生时，努力使时钟脉冲传输到锁存器的输入一侧，从而引起翻转。如果 Q 为 1，输入 K 的与门被使能，如果 Q 为 0，输入 J 的与门被使能。电路分析表明对于 JK＝00，01 和 10 的响应与钟控 RS 触发器相同，然而在 11 的情况下，只要条件 J＝K＝CLK＝1，就存在着振荡条件，如图在 7-9 的仿真波形中所示。*

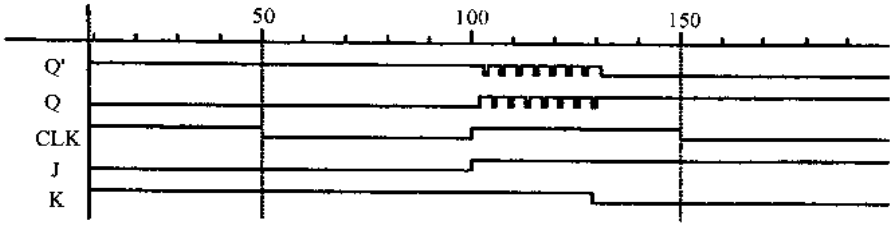


图 7-9

为了使电路正确工作，我们需要提供 1 个非常窄的时钟脉冲或者用延时反馈直到时钟脉冲结束。第 2 种方法较容易且可靠地实现，该实现在图 7-10 所示的主从结构中给出。

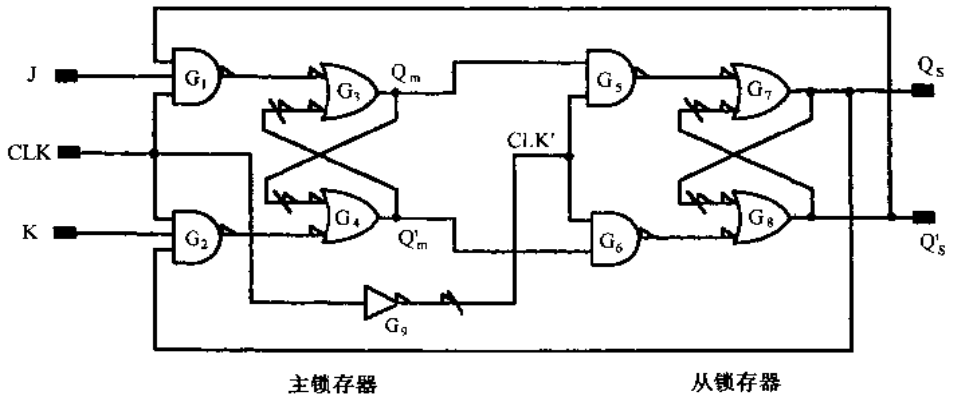


图 7-10 主-从 JK 触发器。黑矩形表示外部管脚

主从触发器包括两个串联的钟控 RS 触发器。提示：加在从触发器上的时钟脉冲相对于加在主触发器上的时钟是取反的，实际上得到 1 个与时钟脉冲宽度相等的负脉冲。

假设输入 J 和 K 为真以及所加的时钟输入也为真。逻辑取反了的时钟 CLK′将为假，并封锁进入门 G_5 和 G_6 的任何信号，因为在这段时间内 Q_s 和 Q_s' 的值不会改变。CLK 信号本身也引入了 G_1 和 G_2(取决于 Q_s 的状态)，并使主锁存器翻转。因为从主锁存器到从锁存器的信号通路被封锁了，所以从锁存器的输出(因此也是反馈信号)不会改变。

当 CLK 返回到逻辑假时(并且相应的 CLK′变为真)，输入门 G_1 和 G_2 被封锁并且不会响应 Q_s 处的变化。现在 G_5 和 G_6 门被使能，主锁存器的输出被传送到了从锁存器。提示：只有从锁存器的输出连到了 IC 芯片的外部引脚上。因此，图 7-10 中的 JK 触发器仅当 CLK 输入发生由真到假的转变时才会改变输出状态。这种同步时钟类型是一种边沿触发形式，其中电路仅响应时钟脉冲触发跳变之前的瞬时的输入数据。取决于设计的是上升沿还是下降沿时钟跳变可引起触发。

钟控 JK 触发器的同步工作总结在表 7.2 中给出。这里，Q_{n+1} 代表紧跟着所加时钟脉冲触发后 Q 的输出状态。从表中可知，当 J 和 K 互补时，Q 等于 J，当 JK＝00 时，Q 保持不变，当 JK＝11 时触发器翻转。

* 如在前面第 6 章脚注中所提到的，具有反馈(如图 7-8 的交互锁存)的电路、模拟有时能遇到问题。参考附录 C。

表 7.2　JK 同步(钟控)操作

J	K	Q_{n+1}	工作方式
0	0	Q_n	禁止
0	1	0	清零
1	0	1	置位
1	1	Q'_n	翻转

7.6　带有预置端和清零端的 JK 触发器

经常希望在同步电路上添加异步(不加时钟)的特性,例如可将它在任何时候复位到特定的状态。考虑改进的 JK 触发器,如图 7-11 所示。这个电路工作于两种方式中的 1 种。

1. 异步:时钟不起作用(CLK = 0)。这种情况下,预置端(PR)和清零端(CLR)输入作为不加时钟的 RS 触发器的置位和复位输入。
2. 同步:PR 和 CLR 不起作用(PR = CLR = 0),时钟有效。

并不是故意将两种方式混合,设计者这样做仅仅是自己担风险。

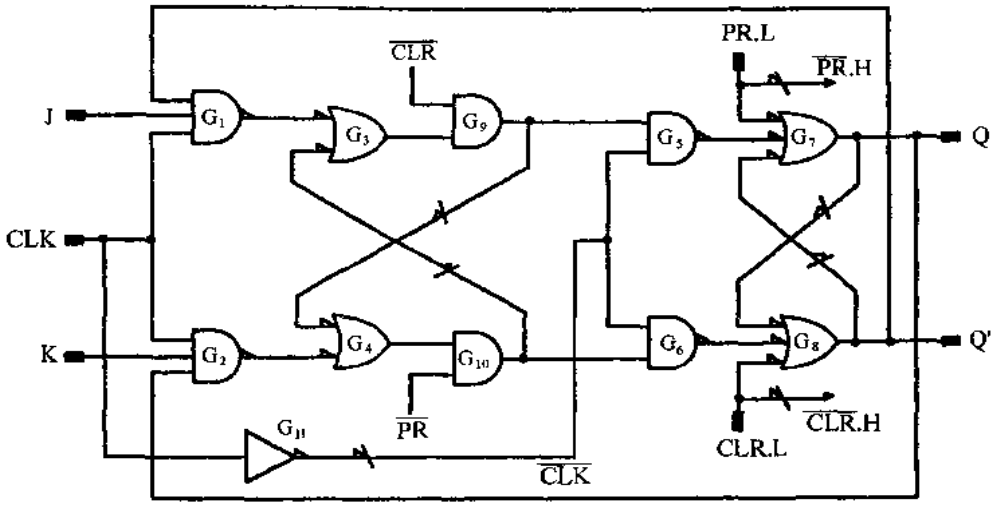


图 7-11　具有预置和清零的 JK 触发器(混合逻辑描述)。
提示 PR.L 和 CLR.L 分别与 PR'.H 和 CLR'.L 相等

带预置和清零输入的主从 JK 触发器的状态表如表 7.3 所示。生产商经常提供带有表示高和低电压的 H 和 L 项的表。逻辑值 0 和 1 的关系取决于硬件的输入和/或输出由 HT 还是 LT 决定。提示:当使带有预置和清零输入的触发器异步工作时,希望时钟保持在逻辑 0。

表 7.3　JK 异步操作

PR	CLR	Q
0	0	Q
0	1	0
1	0	1
1	1	禁止

7.7　触发器中的信号的传输

通过研究主 - 从 JK 触发器的微时序图会得到对其功能的进一步研究。这样的图的结构

在题 6.9 中阐述，在那里具体说明其过程。应该清楚的是随着电路复杂性的增加，利用仿真来研究时序波形变得越来越有优势。

例 7.2 JK 触发器对时钟跳变的响应。假设在图 7 11 中，J = K = 1，PR = CLR = 0，Q_m（G_9 的输出）和 Q 都为 1。开始时，时钟输入 CLK 为 0。在某个任意时刻，时钟变为 1，保持了 50 个时间延时单元，然后变为 0。同时假设在给定的硬件实现中，所有的门都有 5 个时间单位的传输延时。

现在能够确定所有门的初始逻辑值。在时钟跳变发生之前考虑 G_1。其输入为 J = 1，CLK = 0，Q′ = 0。因为逻辑是与，故它的输出一定为 0（根据其硬件，它对应于高电压）。在 G_2 上，输入是 K = 1，CLK = 0，Q = 1，它的输出也是 0。可将分析轮流推广到每个门，直到确定所有内部逻辑电平为止。

当 CLK 到了 1，门 G_1，G_2 和 G_{11}受到影响。在 1 个传输延时间隔过去后，正如逻辑所要求的这些门会改变状态。如第 6.9 题所说明的，过程会继续直到更多的门响应（当然，除非存在不稳定性）。完整的 LogicWorks™仿真如图 7-12 所示。

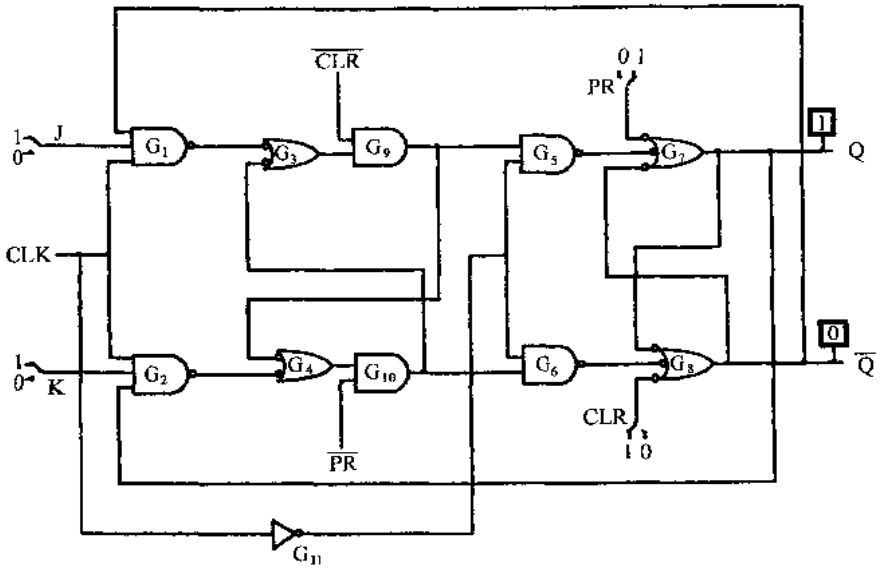


图 7-12(a) 输入设置为翻转模式的 JK触发器原理图

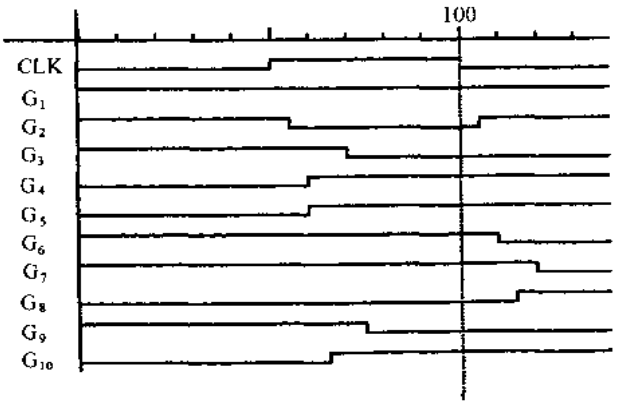


图 7-12(b) 电压波形图

提示：仅仅在时钟脉冲的下跳沿输出发生改变，G_8 在 G_7 的 1 个时间延时前响应。而且有趣的是发现了 G_1 的输出在整个过程保持为高（处于逻辑 0），因为它的所有 3 个输入从来没有全部为真。

7.8 其他触发器类型

第 7.6 节所描述的主从 JK 触发器的状态在时钟脉冲上跳时(0 到 1)J 和 K 输入的信号确定,而输出(从触发器单元的 Q 和 Q′) 直到时钟脉冲的下跳沿才会改变。这种类型的触发器称作负沿触发。正沿触发器件也存在。虽然在数字电子中广泛使用 JK 触发器,但也有其他类型,D 和 T 触发器是最常见的。

D 触发器

这种触发器(D 代表延时)除了它是边沿触发外(非透明),它和第 7.4 节的脉冲选通的数据锁存器相似。它被定义成这样的器件,加了时钟后具有的状态(Q 输出)与时钟脉冲前 D 输入端的信号相等。器件的正向逻辑状态表如表 7.4 所示。表 7.4(a)是同步(PR = CLR = 0)D 触发器,而表 7.4(b)是异步(CK = 0)D 触发器。

表 7.4(a)

D	Q_{n+1}
0	0
1	1

表 7.4(b)

PR	CLR	Q
0	0	Q
0	1	0
1	0	1
1	1	禁止

可将 D 触发器改成在 J 和 K 输入端具有逻辑取反的主从器件。象 JK 触发器一样,可给它配备异步输入 PR 和 CLR(或 S 和 R)。

例 7.3　考虑图 7-13 所示的由 D 触发器组成的电路。假设初始状态所有 Q 输出为 1,并且所有的转换发生在每个时钟的边沿上。随着主 CLR 脉冲的施加,画出 3 个时钟脉冲的时序图。

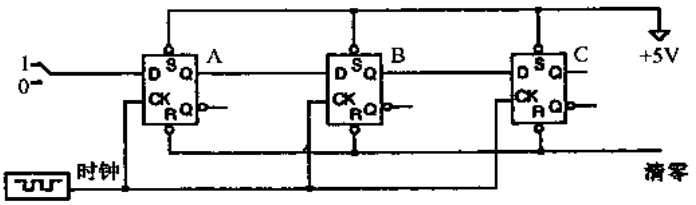


图 7-13

图 7-13 所示的 D 触发器具有 LT 预置(S)和清零输入(R),因此将连接到共同预置线的 5V 理解为逻辑 0 或假(0V 为真)。假设清零线在初始时也处于高电压(逻辑 0),我们从表 7.4 中可知,如果 CLR 变为 0V(逻辑 1),输出将达到 0 而且当 CLR 返回到逻辑 0 时它保持不变(直到 0 到 1 的时钟跳变发生)。提示:因为输出是 HT,所以逻辑 0 用 0V 表示。

紧跟着清零信号的第 1 个上升时钟跳变会使触发器 A 变为 1,因为它的输入 D 为 1。因为其他两个触发器的 D 输出端为 0,所以它们不会响应。第 2 个时钟脉冲会使 B 到达 1,因为现在它连在 A 的 D 输出为 1。同样的,输出 C 在第 3 个时钟脉冲时变为 1,而且不会发生进一步的转变。

所期望的 LogicWorks™仿真波形如图 7-14 所示。提示:考虑了小的传输延时影响。

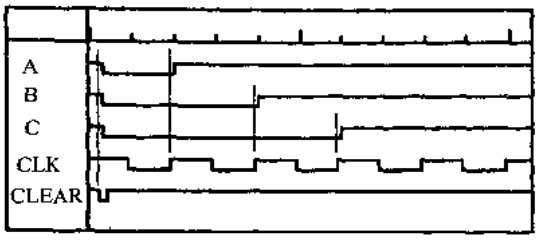


图 7-14 电压波形

T 触发器

这种触发器(T 代表翻转)的工作原理是这样的,当它的单端输入 T 为 1,输出 Q 会随着每个时钟脉冲跳变。当 T 为 0,时钟没有作用。可将该器件模型化为带有 J 和 K 输入永远连在逻辑 1 上的 JK 触发器。在其他的触发器类型中,T 触发器可能也包含异步清零和预置输入。同步情况的状态表如表 7.5 所示。

表 7.5

T	Q_{n+1}
0	Q_n
1	Q'_n

例 7.4 画出图 7-15 所示的 T 触发器组的宏时序图。假设所有触发器开始时已被清零并且触发发生在每个时钟脉冲的下跳沿。

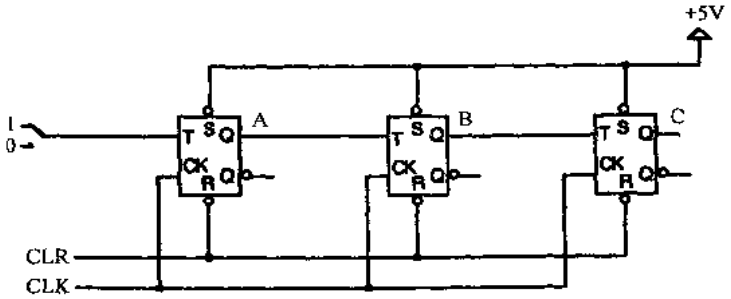


图 7-15

因为触发器 A 有确定的逻辑 1 输入,它会与每个时钟下跳沿同步地改变状态(翻转)。另一方面,当触发器的 B 和 C 输入在下跳的时钟脉冲之前的瞬间为逻辑 1 才会翻转。因此,从图 7-16 中可以看出,触发器 B 接收第 2 个时钟脉冲时它会翻转,因为它来自 A 的输入在触发前的瞬间为 1,而触发器 C 因为此时它来自 B 的输入为 0,所以它不会改变。电路产生的重复序列 ABC=000,100,010,111,000 等。

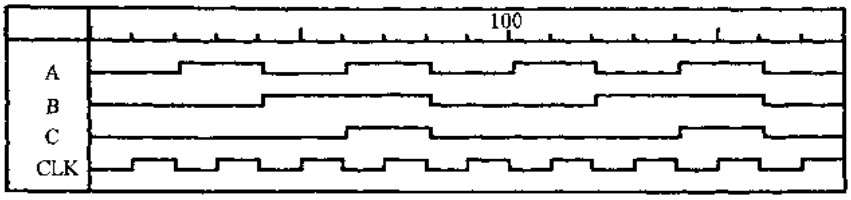


图 7-16

7.9 触发器的触发和时序

与其触发模式无关，对于所有触发器类型，有两个特别要注意的时间。一个是建立时间(t_{su})，它是触发翻转的时钟脉冲沿来临前输入(J、K 等)必须稳定的最小时间。另一个是保持时间(t_h)，它是时钟脉冲触发沿来临后输入必须保持稳定的时间。相对于时钟脉冲，建立时间和保持时间一起建立起输入必须保持不变的间隔。如果这个时间窗内输入改变(如图 7-17 加阴影表示的部分)，触发器生产者不会保证时钟之后器件的最后的状态。

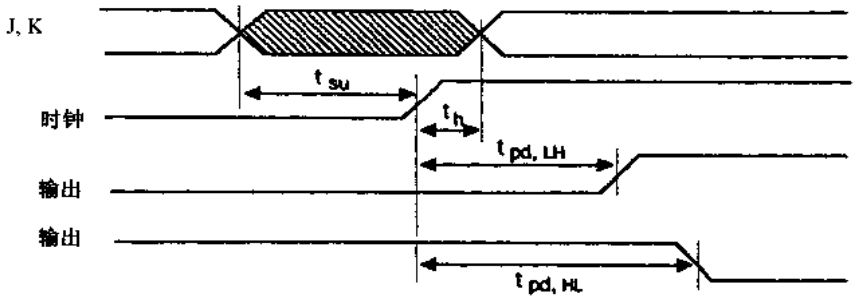


图 7-17

除了建立和保持时间，也具体说明了两个传输延时时间：$t_{pd, HL}$(相对于时钟脉冲，输出高到低跳变的延时)和 $t_{pd, LH}$(相对于时钟脉冲，输出低到高跳变的延时)。允许时钟之后到输出稳定前的这些延时建立时间。图 7-17 表明触发器各种时间参数的关系。

目前可用的逻辑器件，其延时时间范围如上述讨论的那样从几纳秒到几十个纳秒(1 纳秒 $=10^{-9}$秒)。提示：在实际电路中，跳变不会象台阶那样发生，它们有有限的上升时间和下降时间。为简化起见，通常将它们描述成线性的，各种时间的说明是在最小和最大电平的中间测量。

例 7.5　在同步逻辑电路中，经常将触发器阵列(称作寄存器)的输出送进与-或组合逻辑的网络中，用来产生确定寄存器新状态的信号，随着下一个时钟脉冲到来。图 7-18 给出了这样一个系统，对于给定的最坏情况的时序确定最大时钟率。

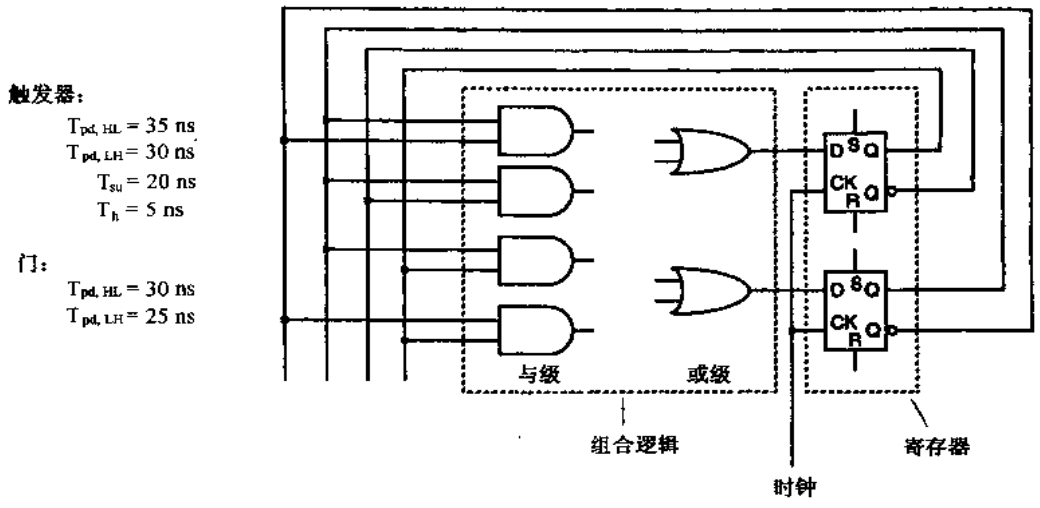


图 7-18

从时钟脉冲的到来开始，考虑时序：

1．因为不知道在触发器输出端将会发生哪种跳变类型，选择最坏的情况(也就是，t_{pd} =

$t_{pd,HL}$=35 ns)。当这段时间过去之后,保证触发器的输出是稳定的。

2. 同样,接在 30 ns 之后,假设触发器输出已通过了逻辑与这一级。

3. 又一个 30 ns 后通过了逻辑或这一级。

4. 在又一个时钟脉冲前,信号在 D 触发器的输入端必须再保持 20ns 的稳定时间。

5. 因为信号在少于 5ns 的时间内不能通过网络,所以要完成保持时间要求。

各种延时在图 7-19 中表示出来。最小的时钟周期是几个单项延时的和,或 115ns。最大时钟率是 $1/115\times10^{-9}=8.7$ MHz。

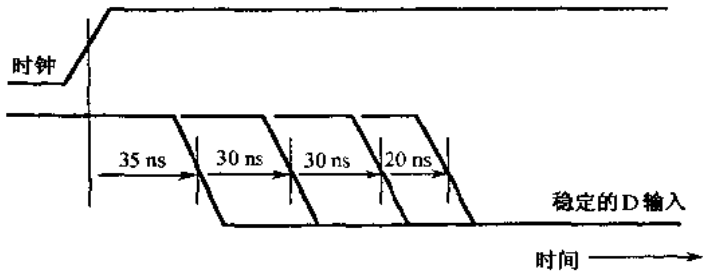


图 7-19

7.10 亚稳度

如果触发器在它的建立或保持时间内遇到有输入信号改变,所谓的亚稳 状态会发生,输出既不是高也不是低,而是位于它们之间的某个电平。在给定的时间内保持亚稳状态的概率,随着时间延时下降,这意味着所有可能的亚稳持续期间,较短的持续最有可能发生。发现表达这种关系的概率密度函数大约为图 7-20 所示的指数。

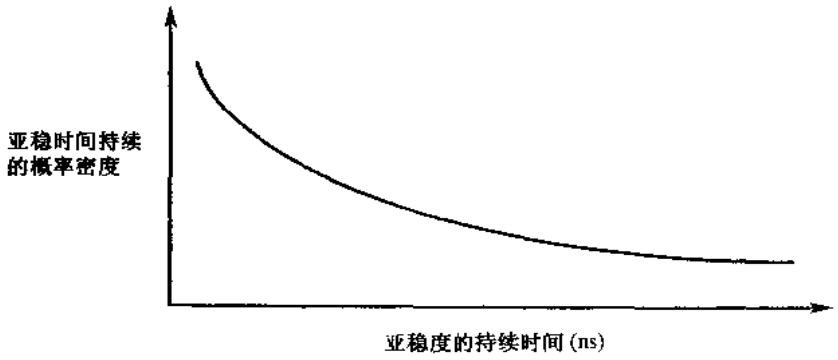


图 7-20

由于亚稳特性的错误可能发生在高速电路中,这里,希望一个触发器的输出在它本身被触发之后立即触发另一个触发器,或者加在触发器上的数据是异步的,以至于产生了有限的可能性,也就是输入对于时钟脉冲而言变化太快。如果要求触发器的输出稳定的时候而它却处于亚稳状态,便会发生错误。

亚稳特性用平均故障间隔时间(MTBF)来具体说明:

$$\mathrm{MTBF}=\frac{1}{F_{clk}\times F_{data}\times T_W\times\exp(-T_R\Delta t)} \tag{7.1}$$

F_{clk}是触发器的时钟频率,F_{data}是数据输入信号(D, J, K, …)的时钟频率,T_W 和 T_R 是常数,取决于在制造触发器中使用的技术,Δt 是时钟脉冲触发边沿到使触发器输出的连续被选通的脉冲之间的时间延时。大多数情况下,用单个时钟脉冲源的连续脉冲来提供触发器触发

和数据选通，因此 Δt 与周期 $1/F_{clk}$ 相等。

例 7.6　给 D 触发器施加 10MHz 的时钟，并具有以大约 10MHz 的速率改变的非同步的输入数据。对于所采用的技术，$T_W = 0.5$，$T_R = 0.75 \times 10^9 s^{-1}$，确定此时的 MTBF。对于 20MHz 的时钟率重复计算。

这里假设是时钟边沿触发器，希望它在下一个触发时钟边沿到来时有稳定输出。因此 $\Delta t = 1/F_{clk}$。

$$F_{clk} = 10^7 s^{-1}$$
$$F_{data} = 10^6 s^{-1}$$
$$T_W = 0.5 s$$
$$T_R = 0.75 \times 10^9 s^{-1}$$
$$\Delta t = 10^{-7} s$$

利用公式(7.1)，计算 $MTBF = 7.47 \times 10^{19} s$。因为它如此大，我们肯定不会活那么长看到失败这一事实，因而有所安慰。

假设时钟频率现在加倍为 $2 \times 10^7 s$。并考虑了含有指数 Δt 的非常大影响。重新计算得到 MTBF = 1932 s(32 min)，这确实是很重要的！时钟脉冲太小不允许正常工作，在本质上电路是无用的。

MTBF 方程表明如果能够改变芯片制作过程来给常数 T_R 的值加倍，那么升高了时钟频率的有害影响几乎可完全克服。若 $T_R = 1.5 \times 10^9$ 和 $F_{clk} = 2 \times 10^7 s^{-1}$，那么 $MTBF = 3.73 \times 10^{19} s$。

因为 T_W 和输入数据率没有出现在指数中，我们知道只要是关于亚稳态错误，它们相对于 T_R 和时速钟率而言并不是很重要的。

习 题 解 答

7.1　生产商制造图 7-21 所示的与非硬件电路实现钟控 RS 触发器。指出在输出端 Q 和 Q′ 从高到低跳变的传输延时($t_{pd,HL}$)比从低到高跳变的传输延时($t_{pd,LH}$)长。构建微时序图并说明是正确的。

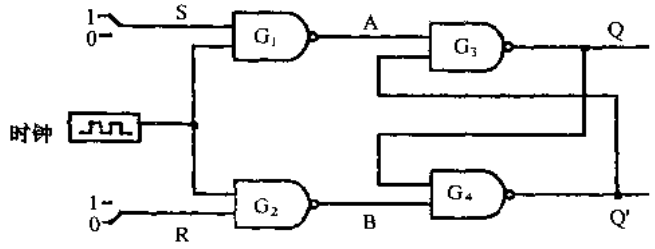


图 7-21

解　图 7-21 是正向逻辑硬件图，通过把它转化成纯粹的逻辑形式可以很方便地分析它。在由互连的与非门组成的电路中，可将每个与非符号认为是紧跟着斜划线(逻辑反)的与门，如图 4-43 的下面的支路中所示的一样(第 4.10 题的第 2 部分)。

考虑电路的置位作用(开关位置如图)，开始时，假设触发器被复位了，也就是 Q = 0(低)，Q′ = 1(高)。当时钟脉冲一出现，在门 G_1 的两个输入端都为 1。一个传输延时之后，它的输出变为 0(低)。这使 G_3 门的输入从 11 变成了 01，因此又一个传输延时之后，Q 从低变为高。这个跳变引起了 G_4 的输入从 00 到 11，又一个传输延时之后，它在 Q′ 上引起了高到低的变化。

对复位作用的相同分析表明，输出从低到高的跳变触发了互补的从高到低的跳变。微时序波形

图如图 7-22 所示，这里对于置位和复位都很明显的是 $t_{pd,LH}$比 $t_{pd,HL}$短，结果是正确的。

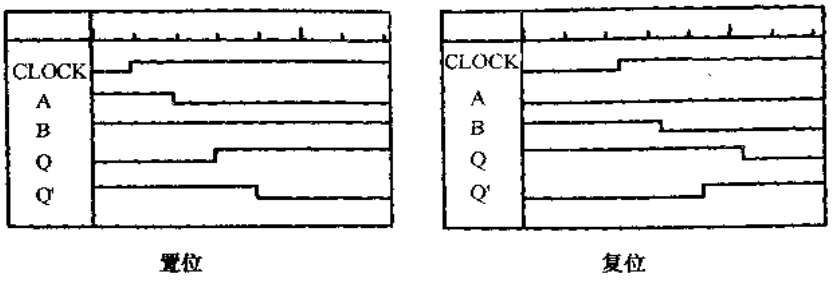


图 7-22

7.2 设计题 7.1 电路的最初尝试如图 7-23 所示，其中 G_1 门和 G_2 门是与门而不是与非门。电路不能工作。确定发生了什么错误并画出微时序图。

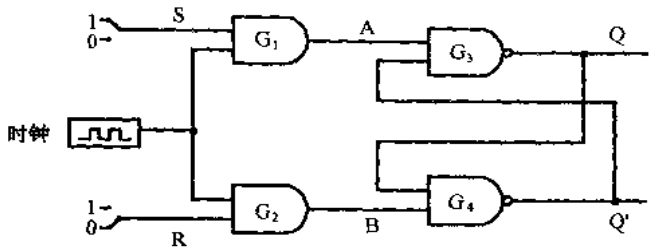


图 7-23

解 在这种情况下，每当时钟为低时，因为门 G_1 和 G_2 的输出都为低，所以 Q 和 Q′门的输入都包含 0，因此，Q 和 Q′都将为高。这是 RS 触发器不允许的状态。进一步分析，当时钟变为高时，电路不能正常工作。在两个时间延时之后输出端 Q 将变为低，而因为 Q′被 G_2 门的输出保持为低，它将保持不变。因此，在 1 个正的时钟脉冲期间，输出状态将恰恰是所期望的反并且它不会锁存。微时序图如图 7-24 所示。

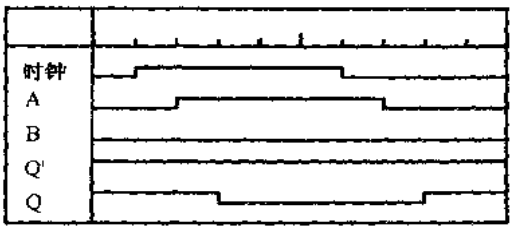


图 7-24

7.3 一学生设计了图 7-25 所示的完成钟控 RS 触发器的电路，希望它如表 7.1 所表示的那样工作。如果用 7.1 题表示的传输延时之间的关系，确定电路是否会正常工作。

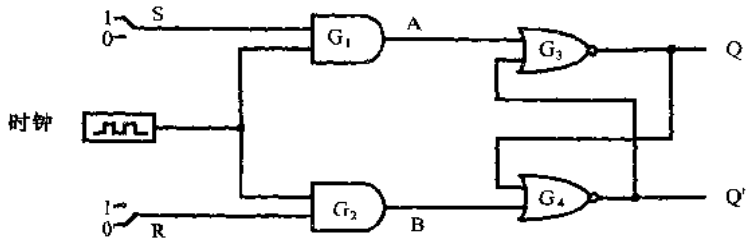


图 7-25

解 如电路所示，正跳变的时钟脉冲会通过 G_1 门，在 G_3 门上面的输入端产生逻辑 1。如果 G_3 门的两输入之前都是低，这意味着为了使 Q 改变，时钟脉冲前 Q′必须为低，这个变化才会有影响。因此，当 S=1 和 R=0 时时钟脉冲会使触发器复位(使 Q 变为 0)。当把开关设置为反向时，时钟会将触

发器置位(使 Q 变为 1)。

提示:这两种情况下,00 输入与 01 或 10 输出条件兼容,意味着电路能保持为复位或置位状态;也就是它能锁存。这里仅存在一个小问题:为了把它分类为 RS 触发器,Q 输出一定要与 S 相同而不是它的补,因为它在给定的电路之中。如果相互改变 S 和 R 或 Q 和 Q'的设计,可很容易的将这种情况改正。

置位和复位输入条件的微时序图如图 7-26 所示。看到从高到低的输出跳变在从低到高的跳变之前,因此相关的传输延时时间与第 7.4 题表明的相反。

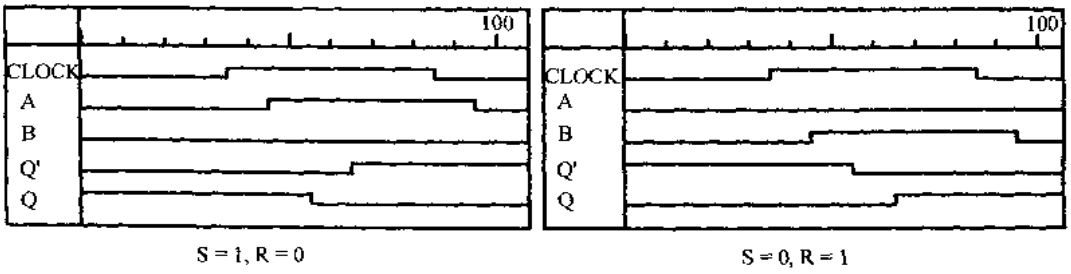


图 7-26

7.4　画出第 7.1 题 RS 触发器的或非硬件实现。

解　从图 7-27 所示的逻辑图中可以很容易地推导出相应的或非实现(图 7-28)。提示:如果要避免使用其他的硬件,两个输入和两个输出必须为 HT。

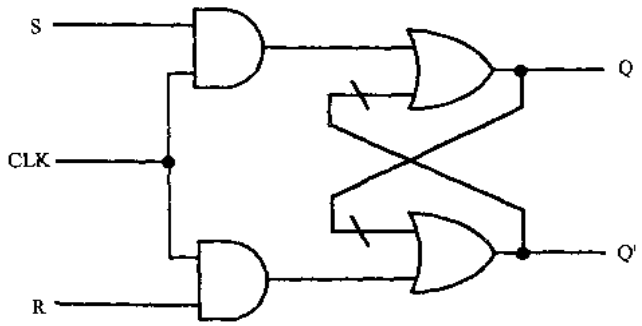


图 7-27

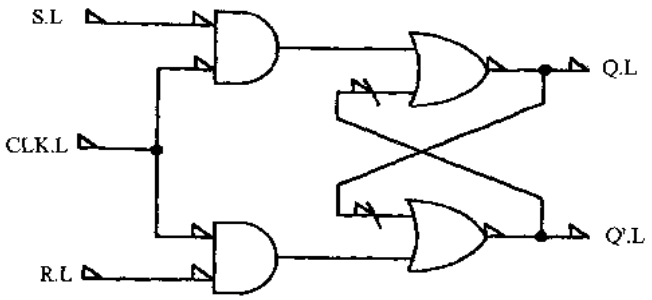


图 7-28

7.5　图 7-29 的电路是一套主锁存器逻辑的与非硬件实现。它的作用是这样的,当 R 和 S 同时为真时,输出 Q 为真。请说明它就是这样的。

解　去消所有半箭头并将电路看作是逻辑图可以很方便地分析它。

第 1 种情况:S=1,R=0。

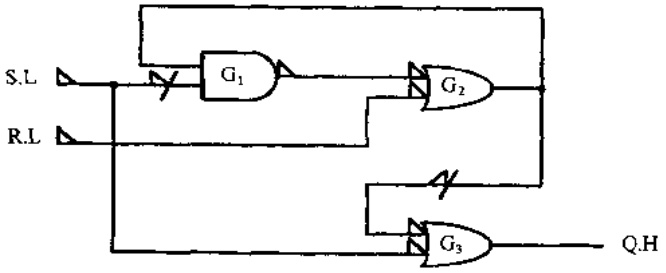


图 7-29

因为 G_3 的输入之一连在为 1 的 S 上，故 Q=1。R 对 Q 没有影响，但因为 G_1 的输出为 0，所以使 G_2 为 0。

第 2 种情况：R 保持为 0；S 变为 0。

因为 G_2 被 G_1 锁存，故 G_2 的输出保持为 0。因此，当 S 变为 0，输出 Q 保持为逻辑 1。

第 3 种情况：R=1，S=0。

只要 S=0，G_2 被 G_1 锁存，$G_2=1$。G_2 后的逻辑取反和到 S 的直接连接使 G_2 的两个输入都为 0；因此 Q=0。

第 4 种情况：S 保持为 0；R 变为 0。

G_2 的输出保持为 1 而且 R 不能影响 G_3。因此，Q 保持为逻辑 1。

第 5 种情况：R=1，S=1。

因为从 S 到 G_3 的直接连接，故 Q=1。这一次，G_2 被 R 保持在逻辑 1 上，因此如果 S 变为 0，Q 会跟着 S 发生变化。提示：但是如果 R 在 S 之前变为 0，G_1 会保持在 0 上，G_2 会变为 0 并且锁存在那里，并且因为逻辑取反，Q 会保持为 1。

如图 7-30 所示的仿真输出表示出典型的特点。

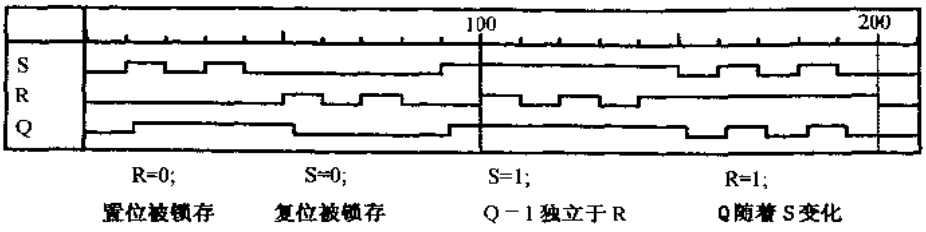


图 7-30 逻辑波形

如要求出 Q′的输出，可以通过添加反相器得到。虽然 RS 触发器比图 7.2 中的锁存器复杂，但是它并没有显示出较简单电路的不确定行为，那样 Q 和 Q′的输出很可能同时等于逻辑 1。经常是这样，性能的改进以牺牲简洁为代价。

7.6 图 7-31 表示了边沿触发 JK 触发器的与非逻辑实现，它正常的功能取决于两个信号路径间传输延时的差别，而不是第 7.5 节说明的主从行为。假设 Q=0，J=1 及 K=0，当施加正跳变时钟脉冲时，画出逻辑微时序图来说明事件发生的顺序。

解 我们主要关心电路的上半部分，因为有来自于 K 的固定的 0 输入的 G_2 门不会改变。

如下确定相关的初始状态：

- 因为 CLK=0 故 $G_1=0$。
- 因为 CLK=0 故 A=0。
- 因为 G_1 的输出被取反，故 B=1。
- $G_7=1$，并且因为取反，锁存器 G_9—G_{10}的输入设置 S 为 0，
- 相同的分析表明，锁存器 G_9—G_{10}的复位输入 R 也为 0，

假设时钟输入上升(CLK→1)。那么 A→1，B→0。提示：因为 CLK 信号仅仅穿过 1 个门而不是 2

个，所以 A 在 B 的 1 个时间延时之前变化。

G_7 输入的序列是 AB = 01, 11, 10。因为在这个序列中它的输入之一总为真，故 G_7 保持为逻辑 1，S 保持为 0，Q 不变。

假设时钟输入下降(CLK→0)。A 又是在 B 之前变化，序列是 AB = 10, 00, 01。这种情况下，G_7 的输入将在 1 个传输延时内为 00，瞬时使 G_7 变为 0，S 变为 1。输出 Q 将变为 1 并通过 G_{10} 锁存在那里。

提示：为了使锁存发生，信号 S 的脉冲持续时间至少与 G_9 和 G_{10} 门的传输延时相等。在实际电路中，可以通过正确地改变门结构来实现。

微时序图如图 7-32 所示。

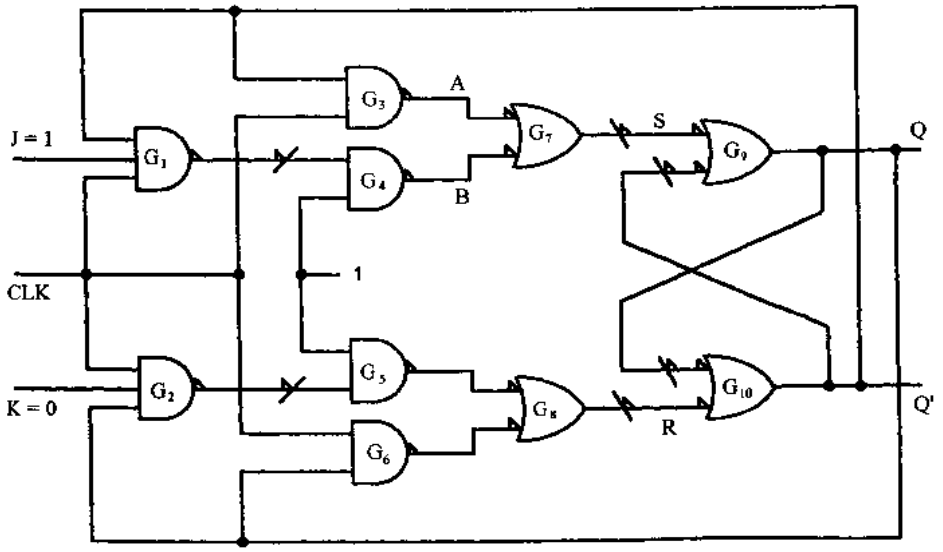


图 7-31

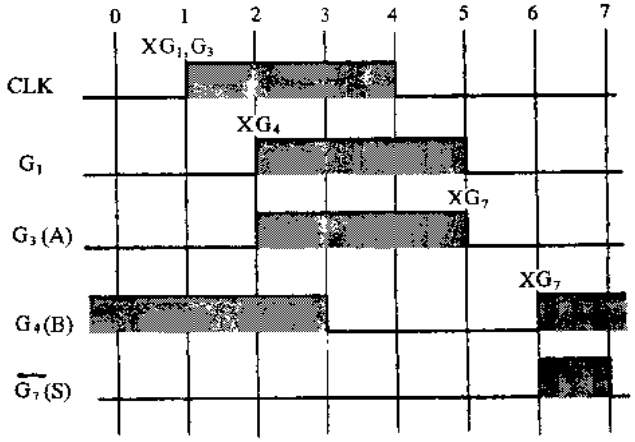


图 7-32

7.7　考虑前面问题的边沿触发器。如果输入变化(这种情况下为 J)及其接近于在时钟脉冲下跳沿前最后可能的 1 个单独传输时间间隔的时刻，那么没有输出响应发生。时序图 7-32说明的就是这种情况。

解　参考图 7-31 和图 7-32 并假设 J 在时钟脉冲下降沿之前 1 个延时时间变为 0。因为为了使 G_1 保持为真，所有输入必须为逻辑 1，故 G_1 会在下一个时间间隔不变。G_3 门不受 J 的影响，但 G_4 会

对 G_1 门的改变做出响应，使信号 B 早一些返回到逻辑 1。可以看到如果这种情况发生，G_7 的 00 输入不会发生，并且因此 S 和 Q 会保持不变。

可在输入 K 上进行极其相似的分析。因此，很明显置位或者复位触发器的指令仅当它保持到时钟脉冲下降沿时才会执行。然而注意，在有限的上升和下降时间的情况中，当输入改变与时钟脉冲有效沿太接近，亚稳状态是可能发生的。

7.8 图 7-33 中的电路由两个在时钟脉冲的下跳沿触发的 T 触发器和一个时钟上升沿触发的 D 触发器，输入初始化为逻辑 1，在每到第 4 个时钟脉冲的下降沿改变它的值。假设在 t=0 时对所有触发器清零，画出表明输出 A，B 和 C 波形图的时序图。

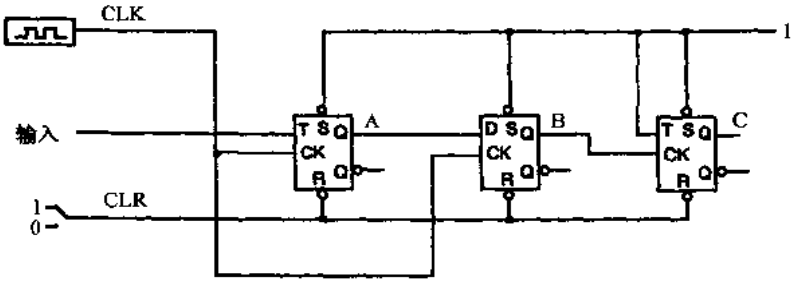


图 7-33

解 当输入为 1 时触发器 A 翻转；否则它不变。当时钟上升时假设触发器 B 为 A 的值。实际上，D 触发器对 A 延时了相当于 1 个时钟脉冲的间隔。每当 B 从 1 变为 0 时触发器 C 翻转，波形如图 7-34 所示，它包括 1 个时间单元的传输延时。



图 7-34

提示：与时钟脉冲下降沿一致的输入变化直到后续的时钟脉冲才会影响触发器 A。当输入信号建立和保持时间为 0 时，仿真非常正确地解释了触发器不会响应这一实事。这个问题的仿真的讨论可在附录 C 中看到。

7.9 两个负跳变 JK 触发器如图 7-35 所示互连。假设两个输入都预置为 1，紧接着施加时钟信号。忽略传输延时，画出 A′和 B 的 5 个时钟脉冲的波形。

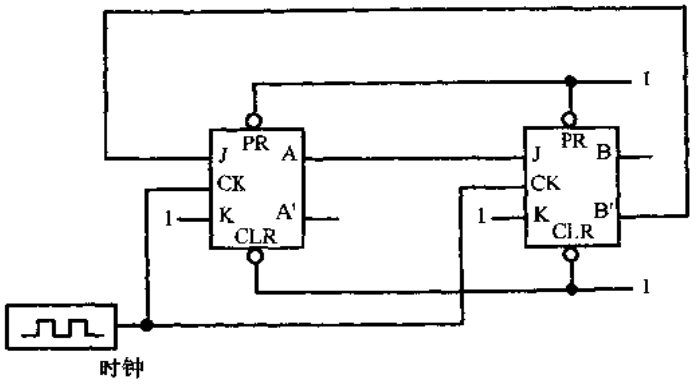


图 7-35

解 ☞ 通过建立时钟脉冲状态表，以有组织的形式处理这种类型的连续问题(见表 7.6)。

表 7.6

			$J_A=$		$J_B=$	
脉冲	A	B	B'	K_A	A	K_B
0	1	1	0	1	1	1
1	0	0	1	1	0	1
2	1	0	1	1	1	1
3	0	1	0	1	0	1
4	0	0	1	1	0	1
5	1	0	1	1	1	1

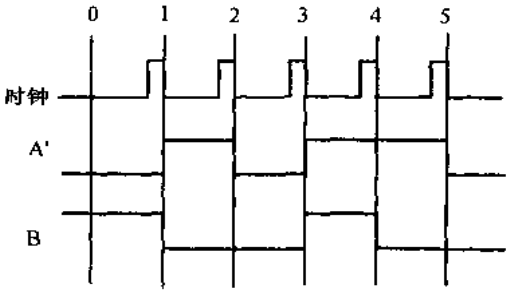


图 7-36

每个触发器的 JK 输入作为指令，告诉触发器当下一个时钟脉冲到来时假设为哪种状态(参考表 7.2)。因为 $J_A=B'$，$K_A=1$，所以给触发器 A 的初始指令是 J_AK_A 为 01，这表明在下一个时钟的下降沿时触发器将被清零($A=0$)。同样，触发器 B 的 JK 输入是 11，这使得触发器翻转(改变状态)。这些指令在表 7.6 的第 1 行被包圈，而且通过指向下一行的项，表明了紧接在第一个时钟脉冲下降沿瞬间的结果。

当第 2 个时钟到达时，希望触发器 A 翻转($J_A=K_A=1$)，并且因为输入 J_AK_A 为 01 而命令 B 到达 0，这是 B 已存在的状态，所以触发器 B 保持不变。

对表每次填入 1 行，利用前 1 行的正确的 JK 状态来确定 A 列和 B 列(对应于触发器输出)中的项。当完成这个过程时，仅仅通过向下读取相应感兴趣的变量的列就可以推知波形。结果如图 7-36 所示。

7.10 图 7-37 表明输入到 JK 触发器的时间变量。设器件初始时清零，画出输出 Q 关于时钟脉冲的波形图。

解 ☞ 正如上一个问题，连续状态的有序列表有助于问题的分析。这种情况下，因为 J 和 K 输入是独立于 Q 的，所以列表形式本质上与 7.2 表中所给出的相同。表 7.7 应是自解释的。

表 7.7

第 n 个时钟脉冲前的输入　　第 n 个时钟脉冲后的输出

n	J	K	Q_{n+1}
1	0	0	0
2	1	0	1
3	0	0	1
4	0	0	1
5	0	1	0
6	0	0	0
7	1	1	1
8	1	1	0
9	1	1	1
10	1	1	0
11	1	1	1

通过向下读状态表的右列，得到了作为时间函数的输出，如图 7-38 所示。提示：时钟脉冲之间输入改变的确切时间对输出结果无影响。这是边缘触发的一个重要特点。

7.11 假设输入 D，S_0，X，S_1 由图 7-39 的电路给出。设触发器响应下降沿触发，并且初始预置($Q=1$)，在 8 个时钟脉冲内画出作为时间函数的输出 Q 波形。

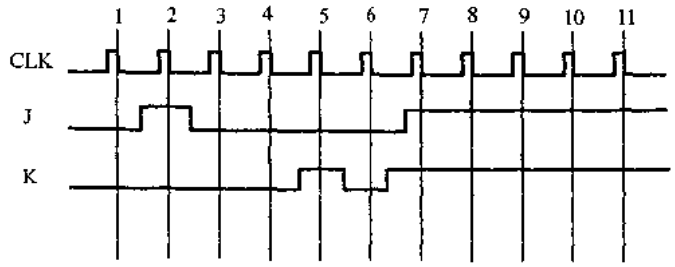


图 7-37

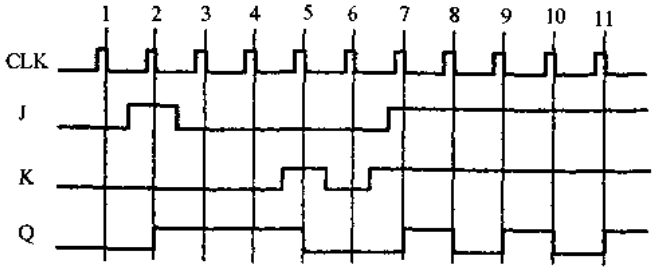
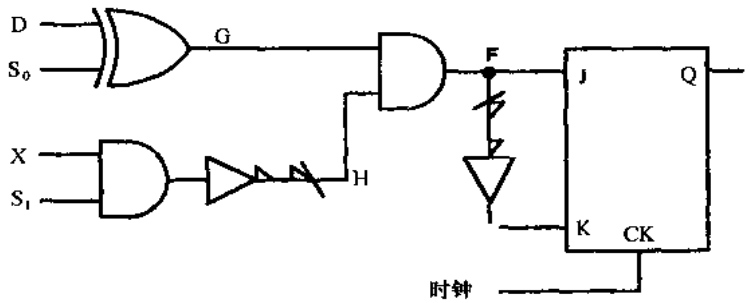


图 7-38



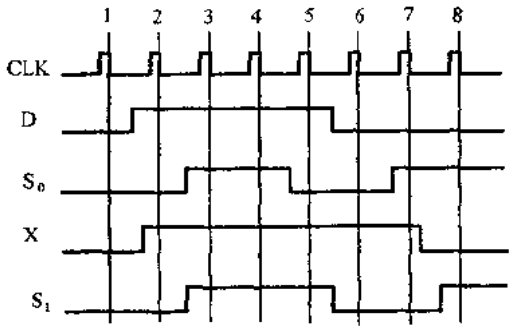


图 7-39

解 J 和 K 之间的输入反相包括具有 D 触发器作用的电路，在每个时钟之后将输入数据送到输出。

从图中可以看到

$$G = D \oplus S_0$$

$$H = \overline{XS_1}$$

$$F = GH$$

通过在图 7-39 的时序图上的任一点画出垂直线，注意输入值，然后利用上面的逻辑方程式，就可确定任何时刻的变量 G，H 和 F 的值。提示：对于电路的组合逻辑部分，变量不受时钟脉冲的影响。另一方面，输出 Q 在每一个时钟脉冲下降沿之前的瞬时与 F 的值相等，并在时钟脉冲之间保持为恒定。

对这个问题，我们将画出中间波形而不用逻辑状态列表(见图 7-40)。

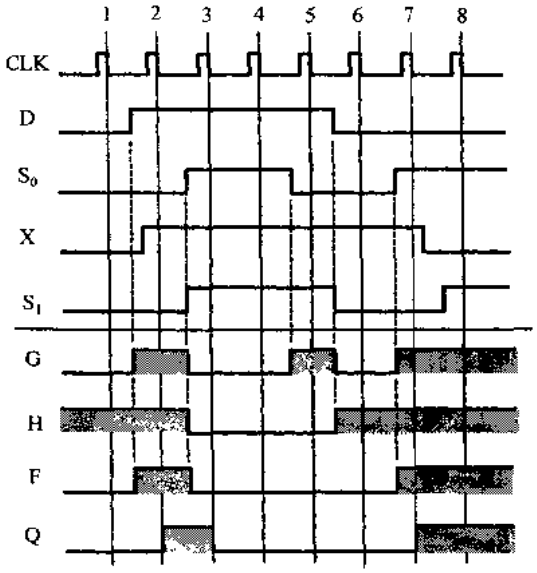


图 7-40

7.12　设触发器在加第 1 个时钟脉冲之前初始化清零，确定图 7-41 所示电路触发器输出端的波形。设触发器由时钟脉冲的上升沿(低到高)触发，并且传输延时小到可以忽略。

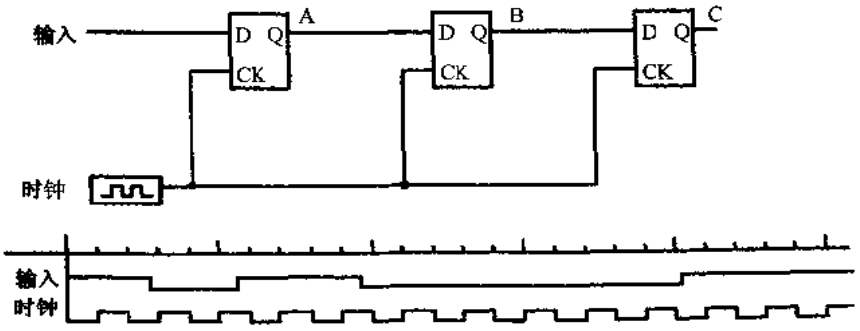


图 7-41

解　触发器被做成 D(延时)类型，因此，在时钟脉冲触发边沿前 D 输入信号，在触发后的瞬间被锁存在 Q 输出端。这个特点在图 7-42 所示的 LogicWorks™仿真中是显而易见的，这里相对波形

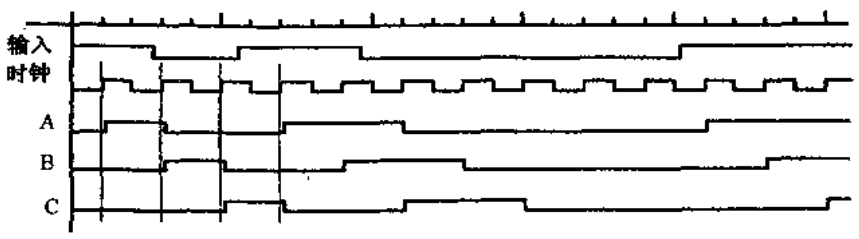


图 7-42

位置清楚地观察电路的延时特性。

7.13 用T触发器替代第7.12题中的D触发器，并画出A，B和C处的波形。又设触发器在加第1个时钟脉冲之前初始化清零，并且是时钟脉冲的上升沿(低到高)触发。使用图7-41所示的同样输入和时钟波形。

解 当输入T为1时T触发器触发(改变状态)并当T=0时保持不变。考虑图7-43波形中对第1个时钟脉冲的响应。因为触发器A的输入在时钟脉冲的上升沿前为1，所以它翻转。因为触发器B的输入A对于时钟稍微延时，在时钟上升的瞬间为0，所以它不变。因为时钟脉冲出现时B为0，故触发器C保持不变。提示：因为它们由相同的时钟信号触发，所以A，B和C波形上的改变彼此相关地排列。

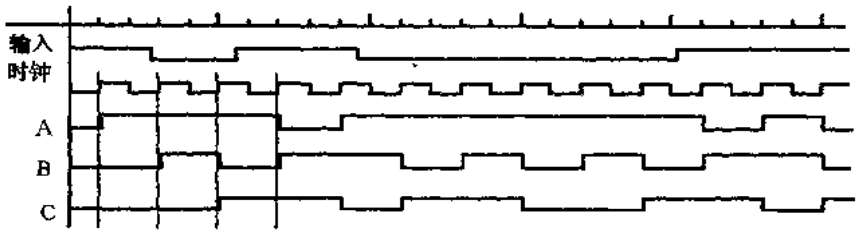


图7-43

7.14 在图7-44中，触发器初始化清零(A=B=C=0)。设A为多路选择器选择线的最高位，触发器为下降沿触发，画出F处7个时钟脉冲的逻辑波形，忽略传输延时。

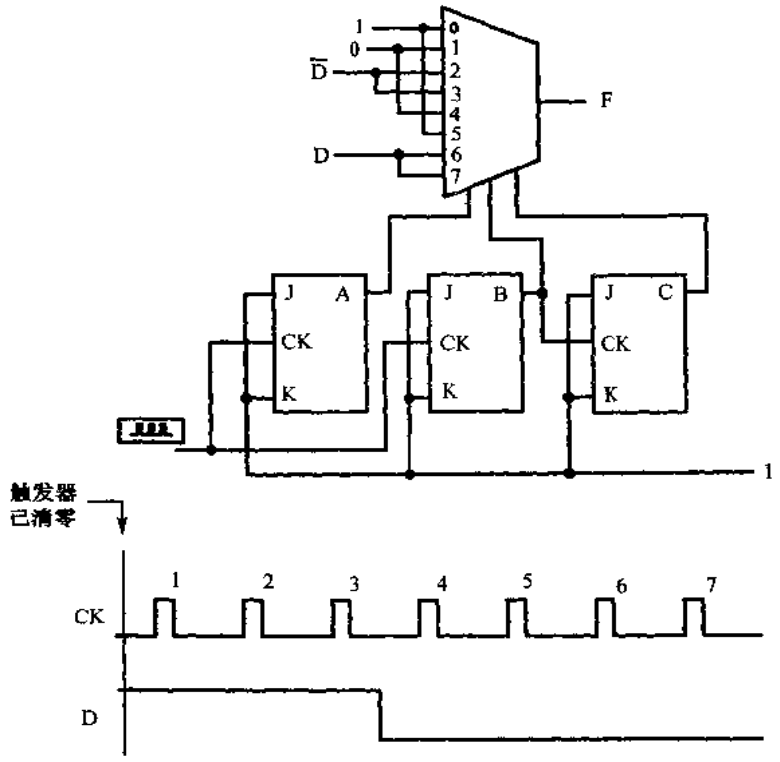


图7-44

解 提示：触发器C的触发来自于B，当J=K=1，当加入时钟时所有触发器会翻转。在第1个时钟脉冲前，ABC=000，这选择了第0根线。因此，初始状态F=1。第1个时钟脉冲使触发器A和B翻转；C不会响应正跳变，因此ABC=110，这选择了第6条线，使F连到D=1上。第2个时钟脉

冲使 A 和 B 翻转;后者使 C 翻转。这时 ABC=001,选择了第 1 条线从而 F=0。第 3 个使 A 和 B 翻转;C 保持不变,ABC=111(第 7 条线)。将 F 连到 D 上,因此,在第 3 个和第 4 个时钟脉冲之间,输出 F 与 D 相同(可看到它发生变化)。连续分析产生了表 7.8 和图 7-45 的波形。

表 7.8

脉冲	A	B	C	线	F
	0	0	0	0	1
1	1	1	0	6	D
2	0	0	1	1	0
3	1	1	1	7	D
4	0	0	0	0	1
5	1	1	0	6	D
6	0	0	1	1	0
7	1	1	1	7	D

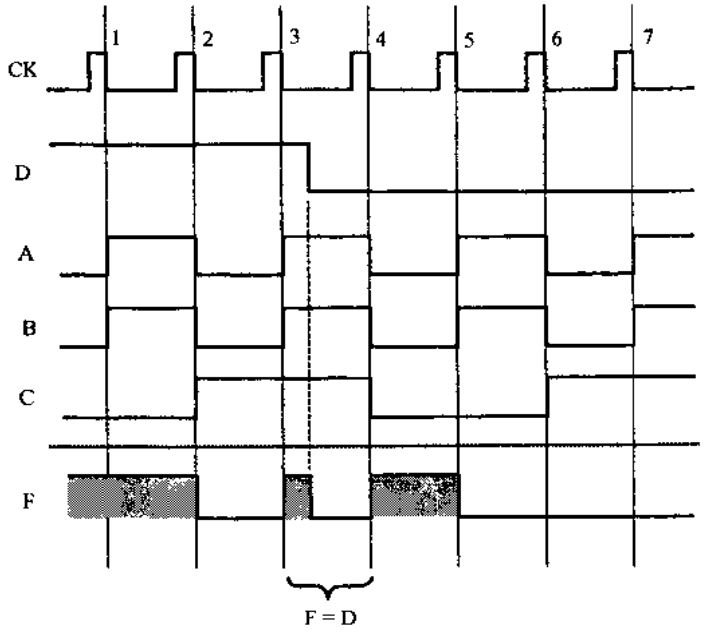


图 7-45

7.15　已知 JK 触发器电路及输入波形如图 7-46 所示,画出输出 Q 的波形。假设触发器在施加时钟脉冲前已被预置(Q=1)并且在时钟脉冲 1→0 的跳变时改变状态。忽略传输延时。

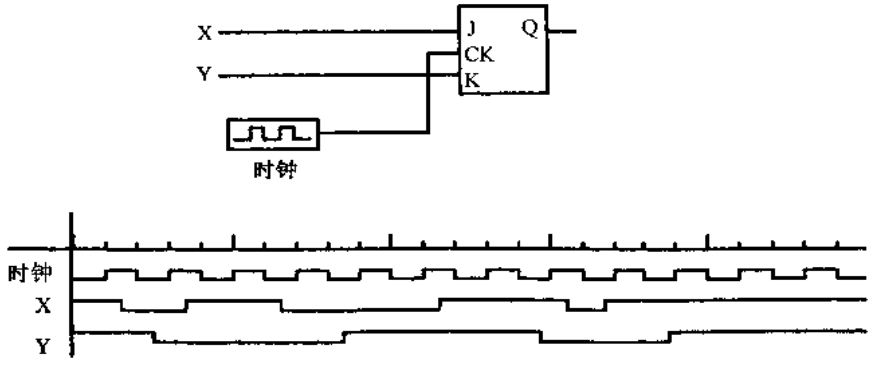


图 7-46

解 回顾同步工作方式下的JK触发器的真值表(表7.2)。

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n'

在每一个恰当的时钟脉冲沿处运用这些规则,产生结果如图7-47所示。

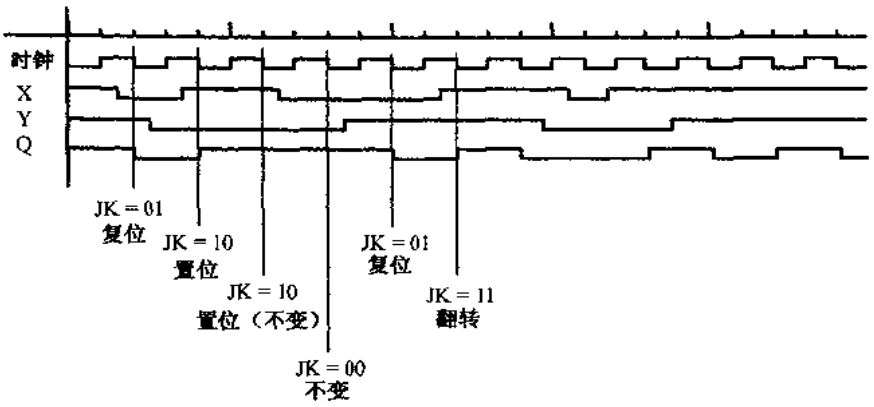


图7-47

补充习题

7.16 要求使图7-4所示的钟控锁存器产生LT输出Q和Q',并对LT信号R,S和CLK响应。确定使用最小数目门的硬件配置。

7.17 RS锁存器的正向逻辑的与非实现如图7-48所示。画出相应的纯逻辑图和状态表。

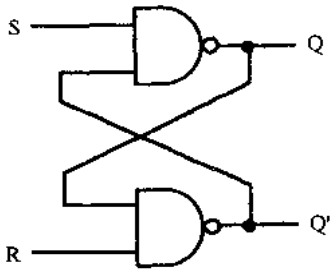


图7-48

7.18 在一个单独的器件上设计综合D触发器和T触发器特点的电路。要求如下:当D和T输入同时为真时,器件不会改变状态;对于所有其他输入组合,器件象正常的D和T触发器一样工作。用表7.2做向导,创建正确的同步状态表。

7.19 考虑图7-49所示的电路。初始时主从触发器都被清零。输入X波形如图所示,假设触发发生在时钟下跳沿,画出变量A和Z的波形。

7.20 一工程师,设计如图7-31所示(第7.6题)边沿触发JK触发器类型,试图通过删去锁存器$G_9 \sim G_{10}$来减少门的数量,导致电路如图7-50所示。

设计者的理由如下:

初始条件:Q=0,J=1,K=0。

时钟上跳。

A点:A→1,B点:B→1。

A点处的变化在B点处的变化之前发生。

G_7的输入序列是AB=01,00,10,而Q保持不变。

时钟上跳。

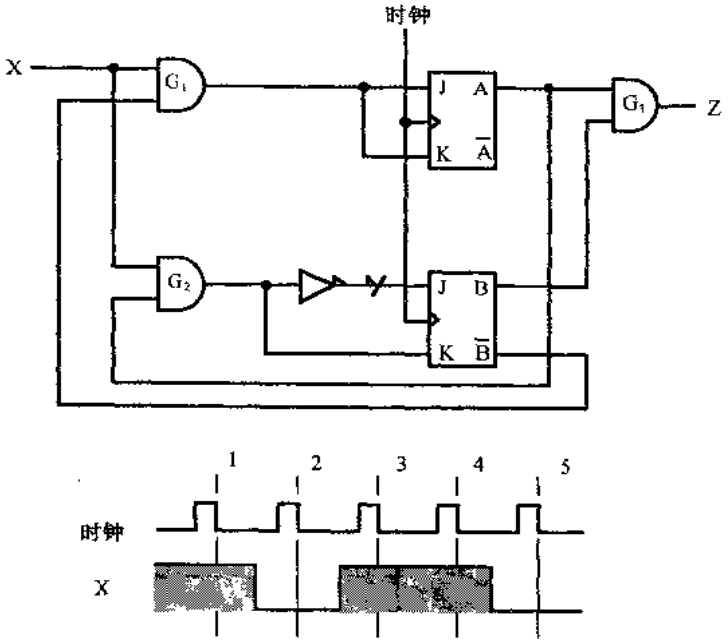


图 7-49

A 又在 B 之前变化，序列是 AB = 10, 00, 01。

在 AB = 00 的主要时间内，Q→1，它被 G_5G_8 和 G_4 锁存。

以上的分析除了一处小的不正确假设外，是正确的！画出微时序图并找出错误。

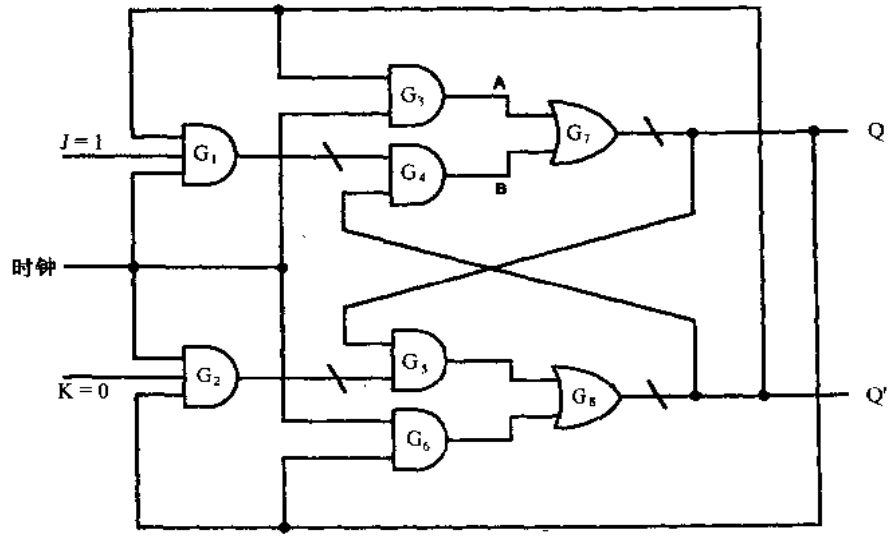


图 7-50

7.21　考虑 7-51 所示的电路。参考第 7.17 题，当时钟为低时，确定输出 Q 和 Q′。

7.22　对于电路 7-51，如果 D = 0，当时钟由 0 变为 1 时将发生什么？

7.23　继续 7.22 题系列，当时钟变为 0 时在 Q 上将发生什么？

7.24　当 D = 1 时的情况重复题 7.22 和题 7.23。

7.25　7.21 到 7.24 题的结果表明 7-51 的电路是正沿触发的 D 触发器电路。当初始值 Q = 1 时，对于图 7-52 所示的条件画出电压宏时序图。

7.26　为了将 D 触发器转换成 JK 触发器，必须添加逻辑，使得当输入 JK = 00 时，为使器件保持原有状态；JK = 11 时，器件翻转。

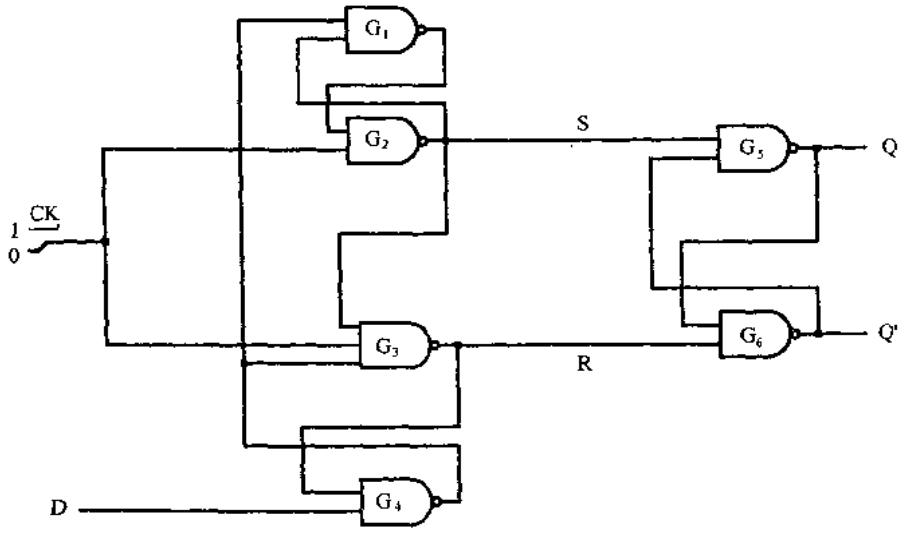


图 7-51

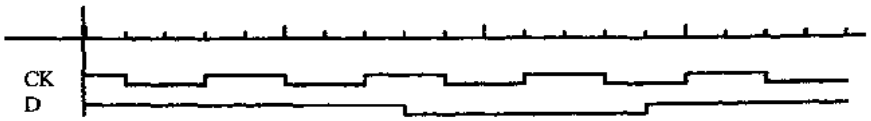


图 7-52

7.27 用户芯片上的门或反相器的组成要求以下的资源分配：

每个门	2 个单元
每个输入(第 1 个之后)	0.5 个单元
每条互连线	0.25 个单元
(包括输入和输出导线)	

如果每个硅资源单元花费为 $0.02，比较单个锁存器(图 7-2)和图 7-11 的 JK 触发器的花费。

7.28 图 7-53 电路中的触发器输出连在多路选择器的选择输入上。这个电路称作扫描器。假设负沿触发，

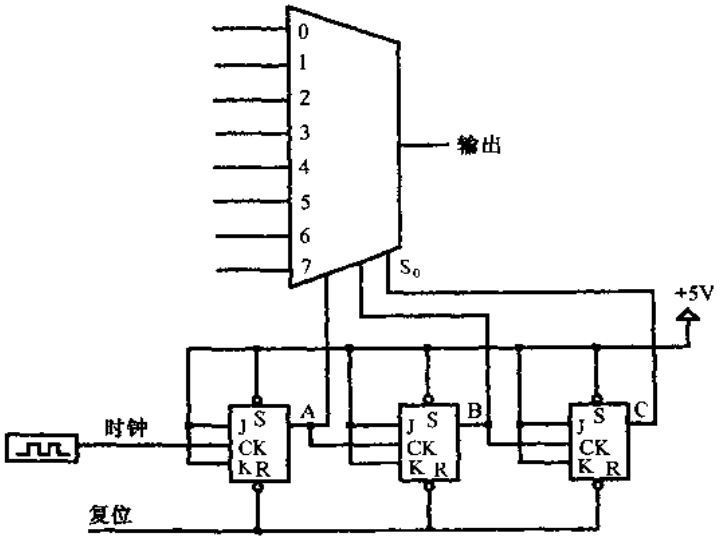


图 7-53

创建信号 A, B 和 C 信号的宏时序图，并说明扫描行为。

7.29 在图 7-12(a)所示的 JK 触发器电路中，通过画出了单个门电压的微时序图来说明预置行为的内部时序。假设初始时 Q = 0，当预置输入变为真(低)时，时钟被封锁(保持为 1)。

7.30 对于图 7-12(a)所示的 JK 触发器电路，假设 J = 0, K = 1，电路初始设置为 Q = 1。当施加单个正跳时钟脉冲时，通过画出单个门电压的微时序图来说明同步清零的内部时序。

7.31 在演示亚稳度的电路中，频率加倍的影响是什么？假设一个时钟源。

7.32 利用例 7.6 的数值来获得第 7.31 题已得到的比率值。

7.33 一学生设计了图 7-54 所示的电路作为 JK 触发器。计划用取反的时钟封锁来自输出 QF 和 QF′的反馈，直到时钟信号返回 0。电路会象要求的那样翻转吗？提示：电路图使用正向逻辑。

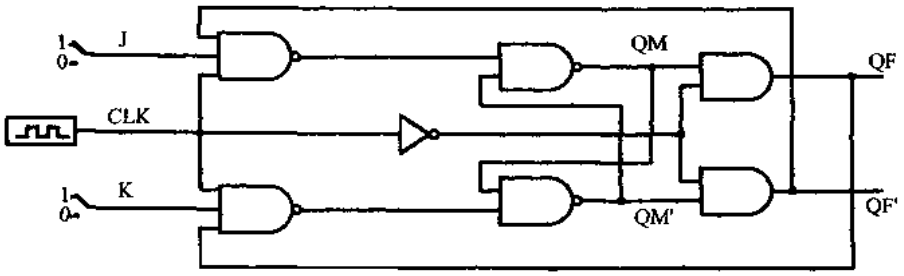


图 7-54

7.34 参考上一个问题，用内部波形时序图来确定简单的定位使学生的电路满足设计要求。

7.35 图 7-55 所示的电路能将 D 触发器转变成 T 触发器吗？用时序图来说明你的结果。

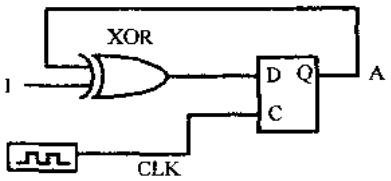


图 7-55

7.36 设计将 T 触发器转变成 D 触发器的电路。

7.37 用例 7.5 的技术，用多路转换器替代组合逻辑中使用的与非门，求出图 7-18 所示电路类型的最大时钟率，假设器件的时间参数如下：

器件	Max, ns	Min, ns
触发器：	$t_{pd,HL} = 20$	$t_{pd,HL} = 10$
	$t_{pd,LH} = 25$	$t_{pd,LH} = 15$
	$t_{su} = 10$	$t_{su} = 5$
	$t_h = 5$	$t_h = 0$
门：	$t_{pd,HL} = 15$	$t_{pd,HL} = 5$
	$t_{pd,LH} = 25$	$t_{pd,LH} = 15$
多路转换器：	$t_{pd,HL} = 25$	$t_{pd,HL} = 15$
	$t_{pd,LH} = 15$	$t_{pd,LH} = 10$

补充习题答案

7.16 用与非硬件如图 7-56 所示。

7.17 逻辑图和状态表如图 7-57 所示。提示：这张状态表是表 7.1 所示电路的逻辑取反。

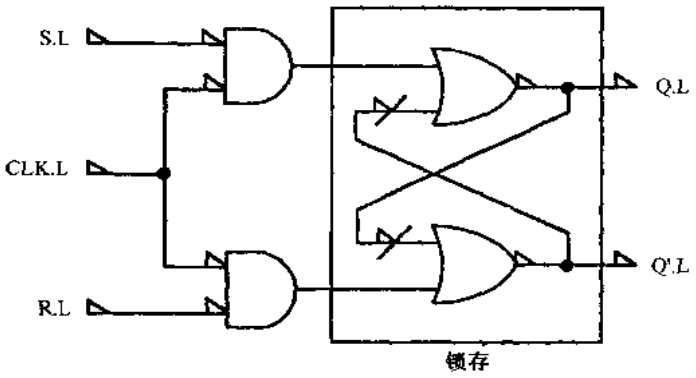
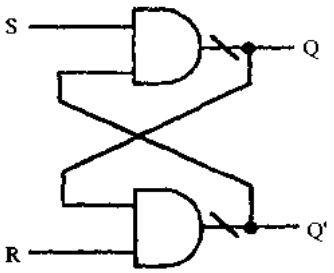


图 7-56



S	R	Q_{n+1}
0	0	*
0	1	1
1	0	0
1	1	Q_n

*两个输出都是 1

图 7-57

7.18　见表 7.9。

表 7.9

T	D	Q_{n+1}
0	0	0
0	1	1
1	0	Q_n'
1	1	Q_n

7.19　见图 7-58。

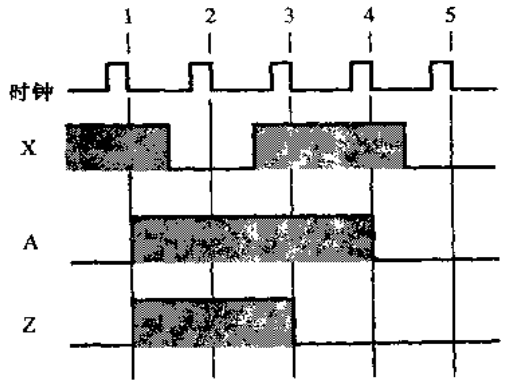


图 7-58

7.20　G_7 的输出如期望由 1 变为 0,但它仅仅保持了 1 个延时间隔;电路不会锁存(参考图 7-32)。

实际上,情况比简单的没有作用更加严重。G_7 脉冲的下降沿影响 G_5 和 G_8,随后它将信号变化回

送到 G_4，等等。交叉连接导致了不稳定性，电路连续振动，如图 7-59 的仿真波形所示。

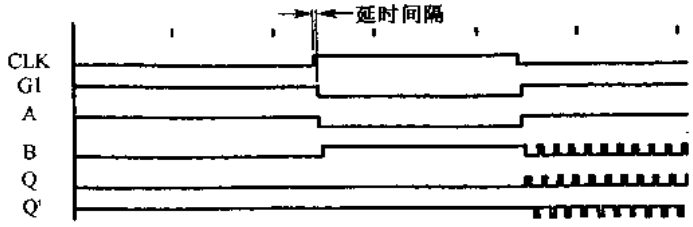


图 7-59

7.21　当时钟为低时，D 没有影响，允许每一个输出状态。因此 Q 的值依赖于历史值。

7.22　Q 变为 0，或如果这是它的最终状态它将保持在 0 处。

7.23　Q 保持为 0。

7.24　Q 变为 1，或如果这是它的最终状态它将保持在 1 处。当时钟变为 0 时，它的状态不会改变。

7.25　见图 7-60。

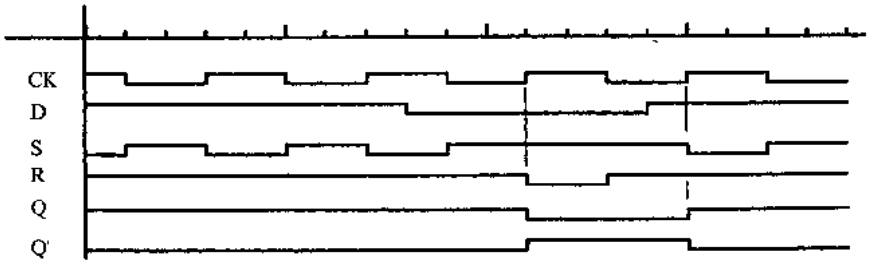


图 7-60

7.26　$D = JQ' + K'Q$。电路如图 7-61 所示。

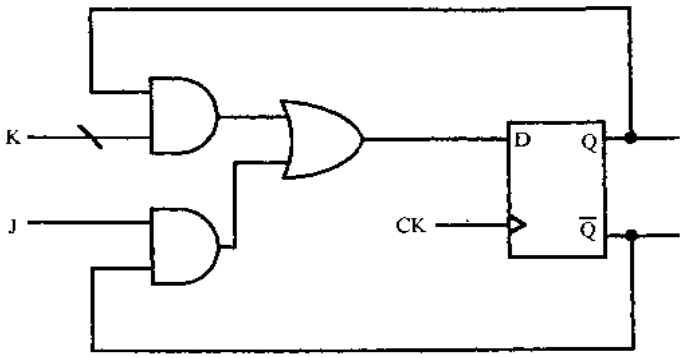


图 7-61

7.27

	锁存器	触发器
资源：	6.5（$0.13）	36.25（$0.73）

注意：JK 触发器比简单的锁存器能够完成更加复杂的操作，这是很重要的。

7.28　多路转换器的输入线将按以下的顺序选择：0-4-2-6-1-5-3-7-0。时序图如图 7-62 所示。

7.29　见图 7-63。

7.30　见图 7-64。

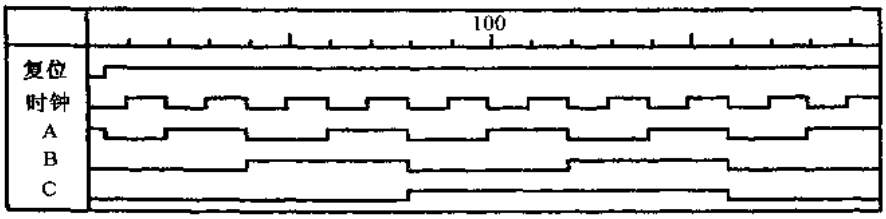


图 7-62 为说明复位,仿真中触发器 A 已初始化为高

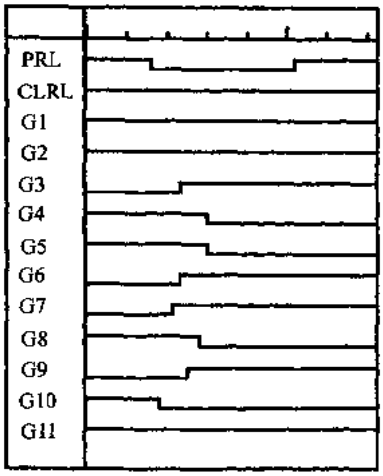


图 7-63

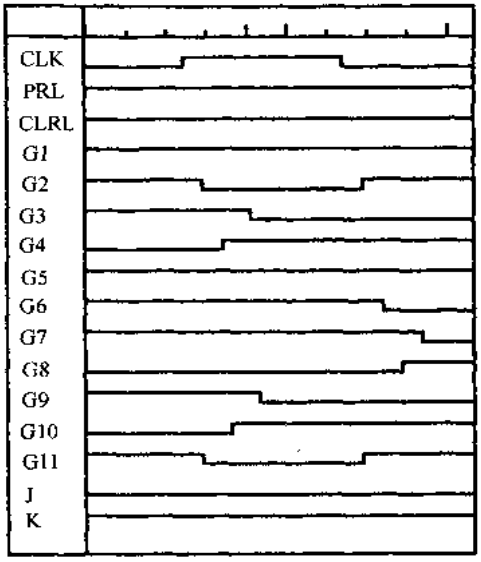


图 7-64

7.31 $\dfrac{\text{MTBF(new)}}{\text{MTBF(old)}} = \dfrac{1}{2}\exp(-T_R/2f_{old})$

7.32 2.59×10^{-17}。

7.33 JK=11 的时序图如图 7-65 所示。当取反的时钟脉冲出现的时候,电路尝试翻转,但输出改变只会持续。



图 7-65

7.34 用 QM 和 QM′作为输出(见图 7-66)。

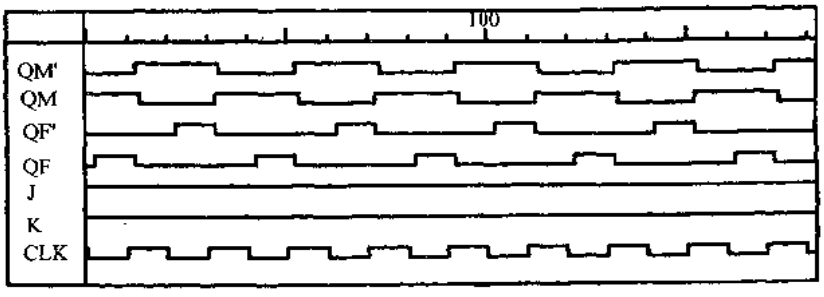


图 7-66

7.35　见图 7-67。



图 7-67

7.36　如果目前的输出值与 D 不同，那么器件会翻转。见图 7-68。

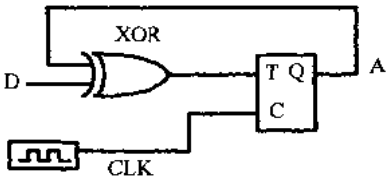


图 7-68

7.37　$f_{max} = 16.67MHz$。

第 8 章　触发器的组合

8.1　寄存器

通常使用共同的时钟脉冲，以某种系统次序安排的触发器组合称之为寄存器。每个单独的触发器称作 1 级，寄存器的状态是触发器输出(1 或 0)的有序排列，通常将其表示为相应的二进制或十六进制数。

例 8.1　确定图 8-1 中所示的以数字排列的寄存器的状态。

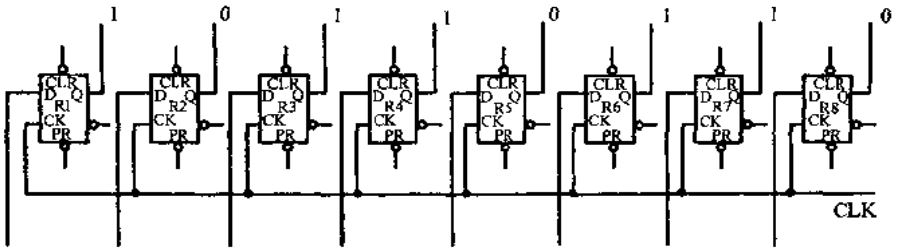


图 8-1　8 位寄存器

设 R_1 级包含最高位，状态为 10110110(二进制)或 B6(十六进制)。提示：寄存器的状态不涉及任何触发器的输入。然而输入的连接确实决定着寄存器从一种状态变为另一种状态的方式。

锁存寄存器

如图 8-1 所示的寄存器具有连到不同数据线的输入，通过给寄存器加时钟可将这些线的集合状态读到寄存器中并存储(锁存)起来。存储的数据会保持不变，直到在寄存器上又加了时钟，此时如果有新的输入数据，它们将被锁存到寄存器中。因此可用一个 n 阶寄存器作为存储或记忆一个 n 位二进制字的存储器件。

例 8.2　将 4 个 D 触发器连在一起作为锁存寄存器，如图 8-2 所示，同时给出了输入信号的宏时序图。假设触发器在时钟上升沿同时转变，在第 1，第 3 和第 4 个时钟脉冲之后表示出寄存器的状态(以十六进制数表示，其中 R_1 为最高位)

在第 1 个时钟脉冲前，A = C = 1，B = D = 0。该数据被时钟上升沿锁存到触发器中，使输出 R_1, R_2, R_3, R_4 等于 1010，它与 A_{hex} 或 $A 相等。在第 3 个时钟脉冲上升沿，所有输出是 1，$R_1, R_2, R_3, R_4 = 1111 = $F。对于第 4 个时钟脉冲，A = 0，B = C = D = 1，寄存器的输出是 0111 = $7。

移位寄存器

将寄存器的级互连成这样一种方式，使表示寄存器状态的二进制数根据命令左移或右移是可能的。考虑图 8-3(a)所示的由 D 触发器组成的 8 级移位寄存器。因为每一级的 D 输入被连在它左边触发器的输出 Q 端，它遵循时钟之后每个触发器的状态将等于它左边邻近触发器时钟前的状态。因此，每个时钟脉冲将使单独的 1 位右移。

如果想到通过将 J 和 K 输入安排为逻辑取反，可使 JK 器件转化为 D 触发器，那么可用 JK 触发器做成移位寄存器(见 7、8 节)。用触发器的 Q 输出驱动 J，同一触发器的 Q′输出驱动 K，就保证了逻辑取反。使用 JK 作为第 4 级移位寄存器如图 8-3(b)所示。

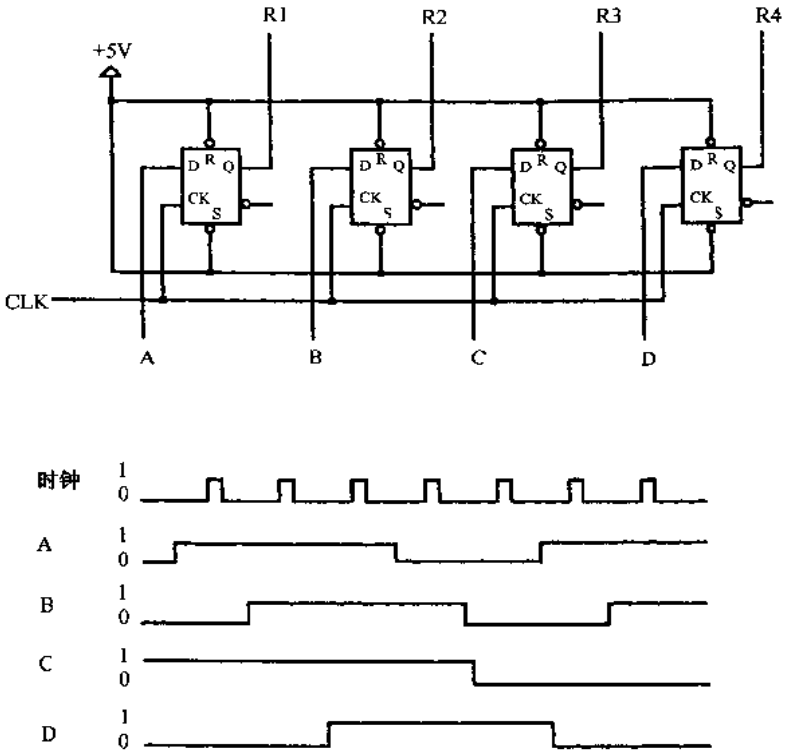
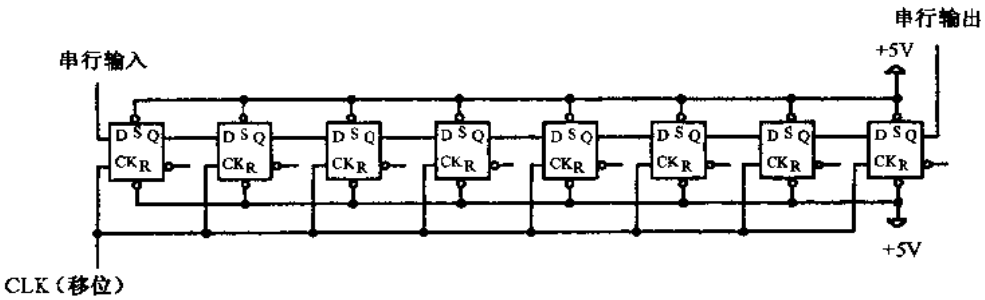
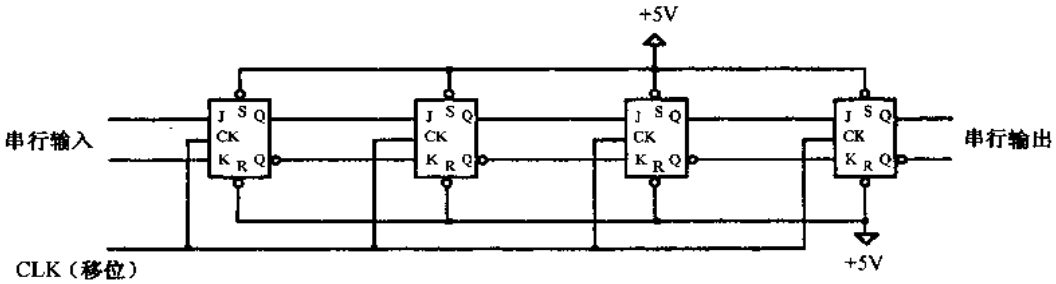


图 8-2



(a) 采用D触发器的8级右移寄存器



(b) 采用JK器件的4级移位寄存器

图 8-3

在这两种情况下，数据出现在左边，它的位与时钟脉冲同步。当移位进行时，数码字 1 次被推送进寄存器 1 位，如同在食堂里将碟子推进到弹簧承载的栈中。碟子的比喻如此恰当以至于将计算机中用作临时存储的移位寄存器称作栈。

例 8.3　用 D 触发器设计可逆的移位寄存器。这个例子说明了将存储器件(触发器)和逻辑器件(逻辑门)组合产生电路的可能性。

移位方向由 D 输入端的连接决定。如果将第 1 级的输出连在第 2 级上，第 2 级的输出连在第 3 级上，如此类推，随着每个时钟脉冲，数据将向右移。另一方面，如果最后一级的输出控制倒数第 2 级，而这级的输出控制前一级等，寄存器会向左移。很明显，需要一种方法能根据命令改变输入的连接。

第 5.2 节中说明的多路开关能有效的实现这一要求，如图 8-4 所示，它描述了 4 级寄存器。用一个二输入多路开关作为每个 D 触发器的输入。我们知道如果将指定为 R/L 的控制信号作为所有多路开关的共同选择端，那么当 R/L＝1(右移)时，每个 D 触发器的输入就被连到最近的前面触发器的输出端，当 R/L＝0(左移)时，连到最近的后面触发器的输出端。

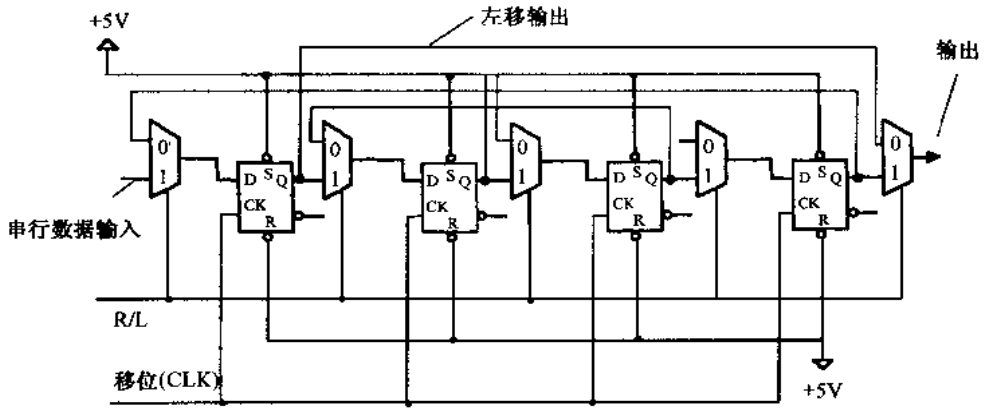


图 8-4　左移连接用黑体示出

8.2　并行-串行转换

如果一特殊的触发器类型具有同步和异步输入，有可能将它用在寄存器中，安排它们完成一种称为并到串转换的数据处理。在图 8-5(a)的系统中，4 位字最初为并行(即所有位同时出现在 4 条不同的线上)。要求以串行方式获得相同数据，这里单个位在 1 条线上连续出现。

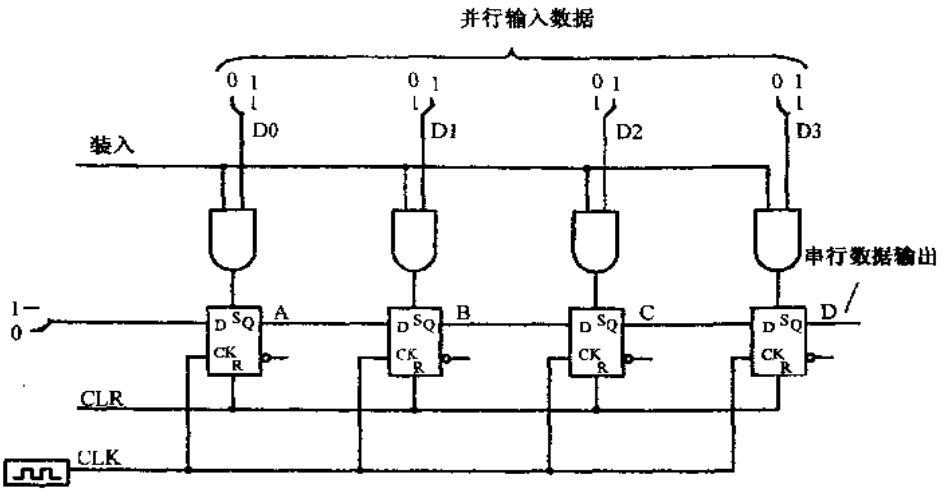
通过参考图 8-5(b)的时序图可以很好地理解转换行为。首先施加异步清零信号，它将寄存器的所有触发器设为 0。接着加 1 个负载时钟，它将通过那些数据位为 1(D_1 和 D_2)的与门和相应的触发器(B 和 C)预置为 1。现在并行数据已被读进寄存器中。

从它们的同步输入点(D)，将触发器连成移位寄存器。当接着施加时钟脉冲时，装入的数据右移，与时钟同步，在 D 触发器的输出端依次出现 1 位。假设以所示的次序施加信号，那么数据将被正确的转换。提示：同步和异步工作方式分开使用，而不是混合在一起。

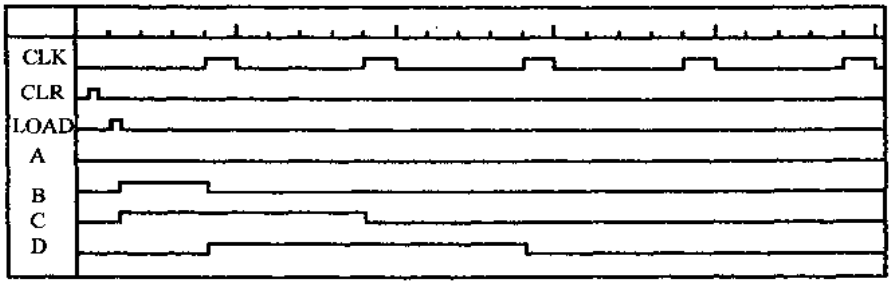
8.3　波纹计数器

将触发器连成某种方式，这使电路能够计算出在输入端的脉冲个数。也就是，看作一组的触发器输出端会表明一个二进制数，它等于从复位时刻到观察时刻出现在输入端的脉冲个数。波纹计数器是计数器的一类，虽然有些慢但设计起来比较简约。典型的例子如图 8-6 所示。

将 JK 输入置为高电平，触发器随着每个时钟脉冲翻转(这种情况下为下降沿翻转)。提示：不象移位寄存器那样，波纹计数器没有共同的时钟连接。相反，只有第 1 个是例外，其他触发器时钟输入由上级触发器的输出驱动。对于每个输入脉冲第 1 个触发器(D)会翻转 1 次；对 D 的每个负跳变下 1 个触发器(C)会翻转 1 次；同样的，C 驱动 B，B 轮流地驱动 A。参考图 8-6(b)相同间隔的时序图，可以确定接着的每个输入脉冲的触发器状态(表 8.1)。如果将输出 A 看作是最高位，电路作为一个 4 位二进制计数器，虽然波形过早的停止，应预测电路可以

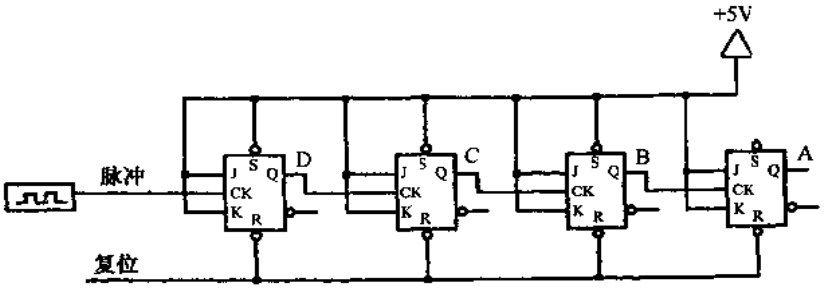


(a) 并－串转换器

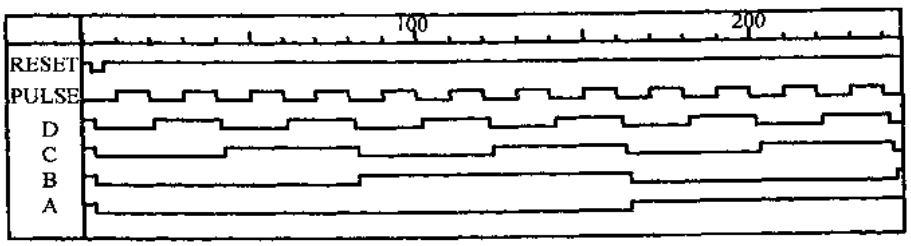


(b) 时序图（正沿触发）

图 8-5



(a) 4级波纹计数器



(b) 波纹计数器时序
（负沿触发）

图 8-6

计数到 15，然后在第 16 个脉冲时循环到 0。很明显，通过添加更多的级可以更多的计数。

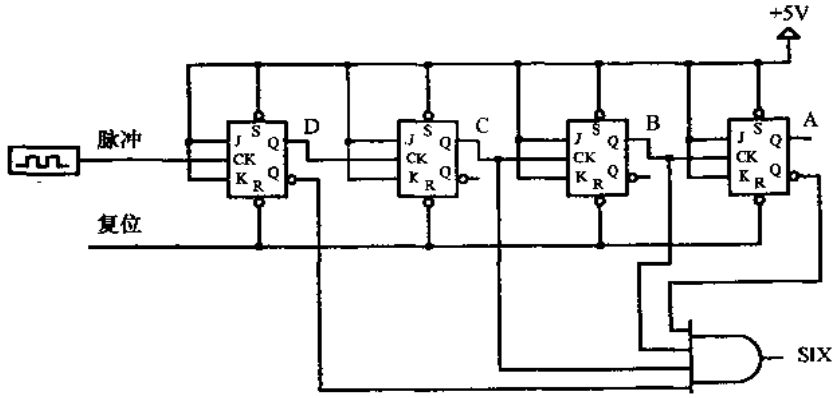
表 8.1

脉冲	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

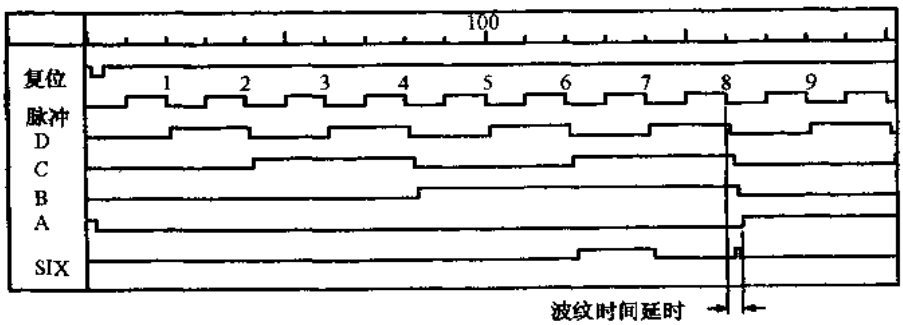
因为每个触发器在它能影响下一个触发器前必须改变状态，经过触发器链的计数过程就像波浪或波纹一样；因此名为“波纹计数器”。

除了作为计数器使用，图 8-6(a)的电路还有另外一个非常有用的特点。从 8-6(b)的波形中可看出从左到右进行，每级的频率恰恰是它前一级的一半。因此可用该电路作为精确分频器。常见的分频器应用是在数字电子表中，它把非常稳定的高频石英振荡器的输出分频，从而获得精确的 1 秒脉冲。

例 8.4　与波纹计数器相关的问题在图 8-7 中示出，其中添加了选通门来标志复位之后



(a) 具有选通门的波纹计数器



(b) 波形

图 8-7

的第 6 个脉冲的出现。其输入为 A′BCD′，仅当计数状态为 0110(十进制数 6)时用来产生输出。

但是，因为与每个触发器相关的不可避免的传输延时(见 7.9 节)，随着触发器的连续触发，从一个计数转到另一个计数的过程中，会出现几个暂态。表 8.2 中表示出从计数 7 到 8 的转变过程中通过的状态序列。当 D 从 1 到 0 改变时，它触发 C 从 1 到 0 的转变，从而轮流地触发了 B，使 A 从 0 变为 1 。稳定状态以黑体表示，而暂态以细体表示，顺时出现的 0110 会引起冲突，见图 8-7b。

表 8.2

计数	A	B	C	D
7	0	1	1	1
6	0	1	1	0
4	0	1	0	0
0	0	0	0	0
8	1	0	0	0

除了易受影响产生冲突外，由于时间积累，波纹计数器还有速度限制。为了使 N 阶计数器正常工作，输入脉冲必须用至少为最大的波纹时间延时$(N-1)T_p$分开，这里 T_p 是触发器的传输时间。因此，波纹计数器很适合对传输带上的器件计数，但是并不适合跟踪高速计算机上移动的脉冲。

8.4　速率乘法器

另一个运用逻辑和存储单元有利结合的例子是速率乘法器。它是产生脉冲链的数字器件，可通过设置控制开关，将脉冲的平均值，即每个单元时间间隔，精确选择成固定的输入脉冲率的一部分。经常将这样的器件用于有效地控制步进电机，它们的轴随着每个脉冲旋转固定的角度。通过选择产生的脉冲数，可获得机械工具、打印机纸的驱动、机器人等的高精度机械定位。

典型速率乘法器的逻辑图如图 8-8(a)所示。这里用 4 位二进制计数器产生在时间上相互关联的 8 个信号(每一级的 Q 和 Q′输出)。这些信号与时钟脉冲及控制输入在 4 个选通与门上合并，只有当满足正确的逻辑条件及输入脉冲出现(真)时，每个门才会产生输出。

参考图 8-8(a)中的电路图，知道只有 C_8 和 A 同时为真时，G_8 门才会通过脉冲；这个条件保持每两个输入脉冲输出 1 次。因此，当 C_8 为真时，有一半输入脉冲出现在 G_8 输出上。同样，只有当 A′，B 和 C_4 同时为真时，G_4 才通过脉冲。这个条件保持每 4 个时钟脉冲 1 次，并且来自 G_4 的脉冲不会与来自 G_8 的任何脉冲同时出现，因为 A 和 A′不能同时为真。同样的方式，G_2 和 G_4 会分别每隔 8 个和每隔 16 个输入脉冲才通过 1 个输入脉冲，并且没有重叠。将选通门的输出通过逻辑或连在一起产生系统输出 F，将其送进功率放大器中来驱动电机。

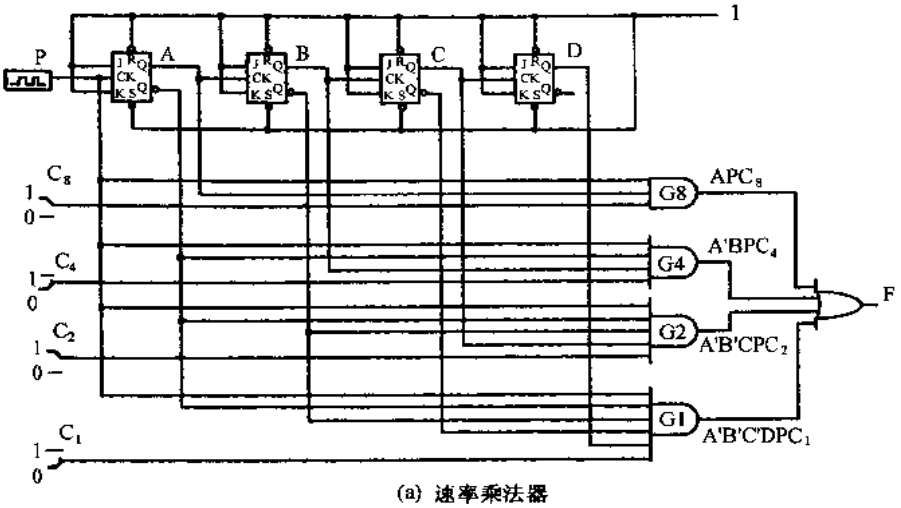
我们知道，通过对控制值的正确选择可以创建输出脉冲率等于输入脉冲率的 0 到 15/16，即以 1/16 为步长的 16 个脉冲率的任何一个。在图 8-8(a)所示的例子中，控制变量 C_8 和 C_2 为真，而 C_4 和 C_1 为假，图 8-8(b)表明输出脉冲间隔并不一致，相反以不规则的速率出现，在第 8.12 题中对其进行定量研究。

将速率乘法器表示为逻辑控制变量的函数的等式可以写作

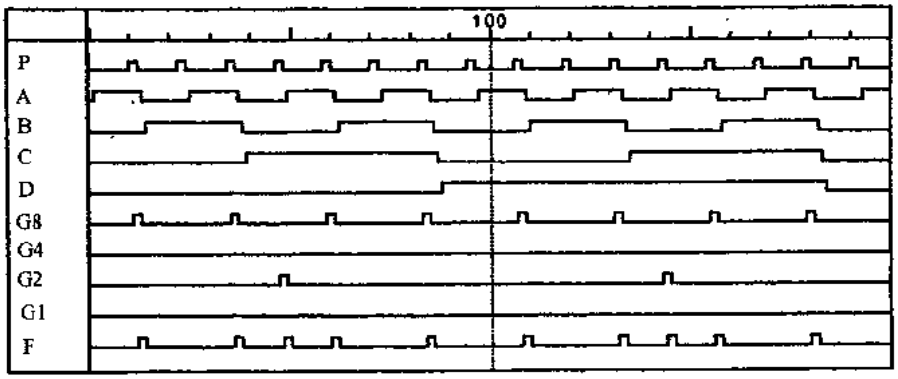
$$M=\frac{1}{2}\times C_8+\frac{1}{4}\times C_4+\frac{1}{8}\times C_2+\frac{1}{16}\times C_1$$

如果需要更好的分步增量，可以添加另外的计数器级数，以及另外的相同数目的选通门和控制输入。例如，为了将分辨率提高到 1/32，就要添加触发器 E，它的 J 和 K 输入连在逻辑 1 上，时钟输入从输出 D 驱动，将添加的选通门连在 A′，B′，C′，D′和 E 以及输入脉冲和新控制线上。控制线由 C_1 到 C_{16}重新标记，速率乘法方程式变成了

$$M=\frac{1}{2}\times C_{16}+\frac{1}{4}\times C_8+\frac{1}{8}\times C_4+\frac{1}{16}\times C_2+\frac{1}{32}\times C_1$$



(a) 速率乘法器



(b) 波形

图 8-8

8.5　随机存取存储器(RAM)

以电子速度改变(写)和读其内容的存储器件命名为 RAM。可能最好的实现是可寻址寄

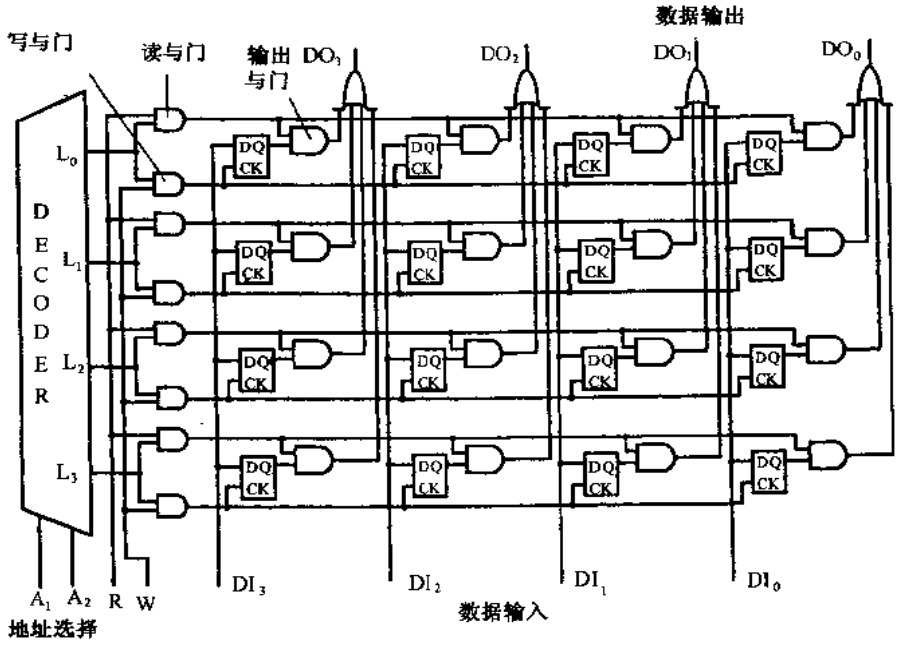


图 8-9　与 RAM 存储器相同的逻辑

存器阵列，每组寄存器都保持一个 n 位二进制字。图 8-9 表明了逻辑上相应的 4 位 RAM 结构，它突出的特点之一是地址译码器，与在 ROM 中使用的功能完全一样，用来选择正确的寄存器。

在这一点上，有一个有序的简短解释。如果所有输入而不是输入之一连在逻辑 1(真)上，我们称与门准备好了。这种情况下，输出会假设为保持在输入的逻辑值，我们可将门认为是哨兵，当它准备好的时候它传递信号，反之它封锁信号。我们知道地址译码器输出选择了被选行上的两个与门，因此如果读(R)信号出现了，读门会产生逻辑真的输出，如果写(W)信号出现了，写门会出现逻辑真的输出。

每一列触发器对应于位这一级(最高位在左边，最低位在右边)。同一列的所有触发器的 D 输入连在一起，如图 8-9 所示，连在一条输入线上。如果施加了写脉冲，它会通过正确的选择好的与门，以时钟驱动数据进入所选(被寻址)行的所有触发器，因此在指定位置存储了一个 n 位二进制字。

在读模式下，连在所选行的每个触发器的输出端的与门被地址译码器选通，因此每一列存储器的位值被传送到大与门上，它具有与 RAM 的可寻址存储空间一样多的输入线。在所有存储的字中，来自与同一级的位(列)的以相同方式连接的与门的输出组成了这一或门的其他输入。因为所选的字(行)，具有选通的与门输出是惟一的，所以每一列到达或门输出的所有输入必须为 0，输出的数据由所选字的位单独决定。

实际上，所有进入或离开存储器的数据都通过输入/输出(I/O)寄存器，如图 8-10 所示，这个寄存器一定与离开芯片的数据输入通信并将数据发送到外面。典型的 I/O 单元如图 8-11 所示，会发现在同样的路径(称作数据总线)上数据进入和离开芯片。通过 D 输入数据将锁存到 I/O 寄存器。当写信号出现时，D 输入连接到数据总线当读信号出现时，D 输入连接到存储器芯片的输出或门上。每个 I/O 寄存器输出位到数据总线的连结是通过称作三态缓冲器的电路，当有读信号对其使能时，它提供了一条导通的路径，没加使能时，它象一个打开的开关。

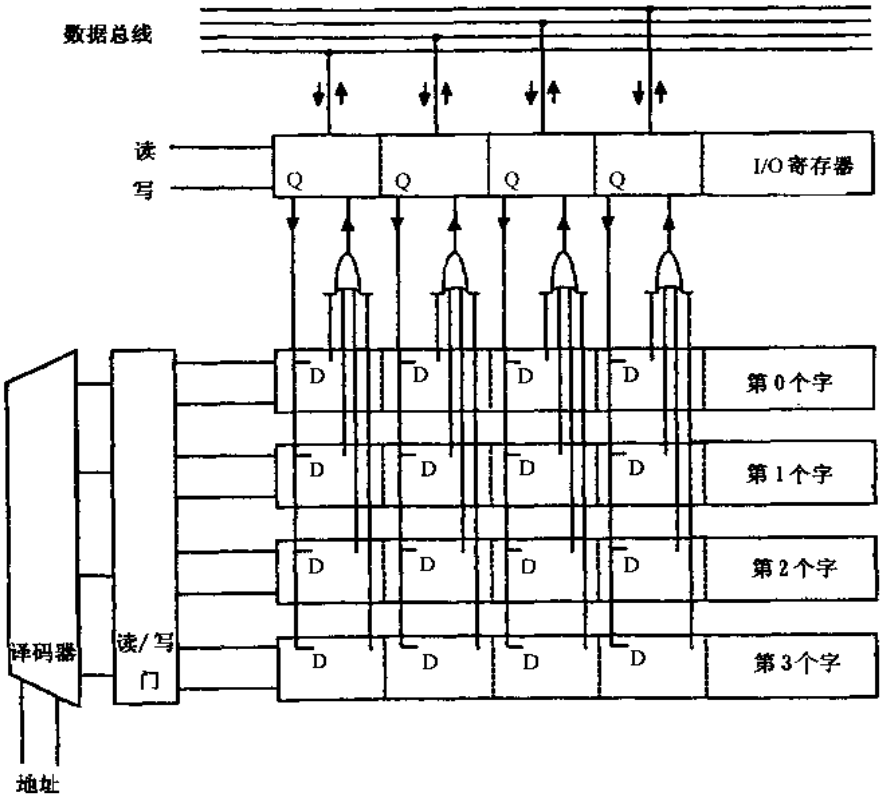


图 8-10　带有 I/O 寄存器的 4 位 RAM 存储器

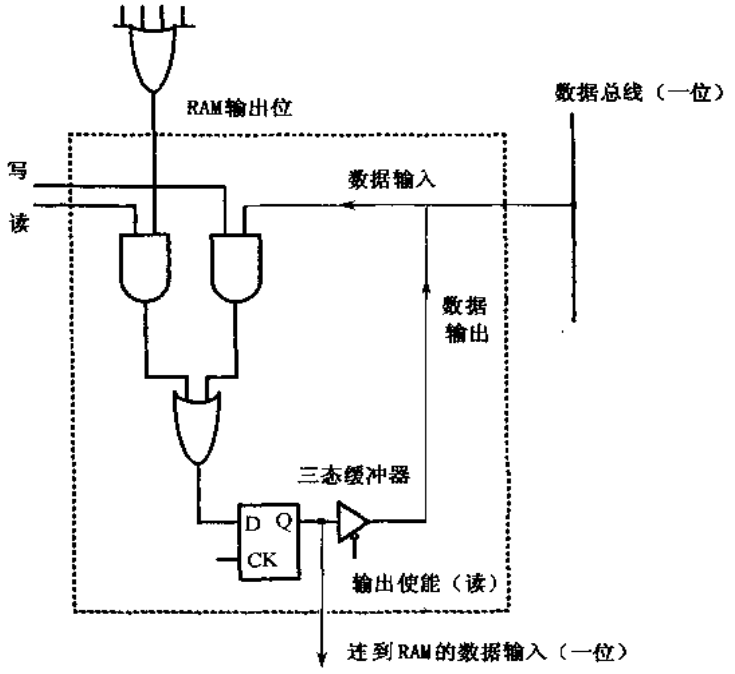


图 8-11　I/O 寄存器单元

通过这种方式，在写操作期间防止 I/O 寄存器的输出电路干扰输入数据是可能的。总之，用作控制开关的三态缓冲器使得不同电路分享共同的数据总线而不相互影响。典型的三态*（TRI-STATE)符号如图 8-12 所示。

在计算机和其他数字系统中，RAM 用于暂时存储数据，一定要取回，以便以后的使用。它的作用非常象手头上计算的便笺本。也可用作存储计算机和/或设备程序，运用特定的操作条件可根据需要修改。这和 ROM 器件形成鲜明对比，ROM 用作永久存储生产商安装进去的计算机指令或数据列。只有在掉电期间才能把芯片取下来，替换它或对它重新编程才能改变存储的信息。

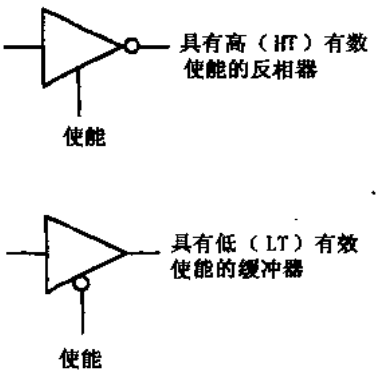


图 8-12　三态器件

习 题 解 答

提示：以下的几个问题涉及到时序电路，要求在接收时钟脉冲前仔细分析状态并对结果输出制成表格。本章讨论的方法适合于相对简单的电路。分析时序电路的通用的系统方法将在第 10 章与状态机的相关的学习中描述。

8.1　连接成环路，在一点上具有逻辑取反的移位寄存器称作麦比乌斯或约翰逊计数器。运用 4 个 D 触发器的串联形式如图 8-13 所示。假设初始化清零并且在时钟上升沿发生跳变，画出每一个输出的宏时序图。

解　当复位时，所有 Q 输出为逻辑 0。D′将为逻辑 1，它将使 A 在第 1 个上升的时钟沿信号时变为 1。作为 B 的数据输入的输出 A，会使 B 触发器在第 2 个时钟脉冲时变为 1。这一改变会传输到 C

* TRI-STATE[2] 是美国国家半导体公司的注册商标。

上，它轮流使 D 在第 4 个时钟脉冲时变化。在这一点上，D'变为 0，使 A 在第 5 个时钟脉冲时变为逻辑 0。现在高到低的跳变会波纹式的通过计数器。直到到达 D，此时，重复上述过程。波形如图 8-14 的仿真输出所示。

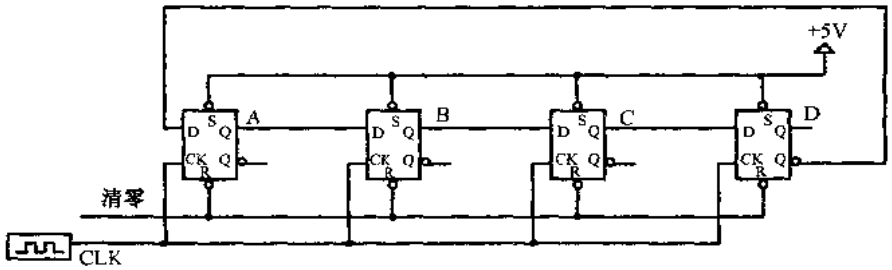
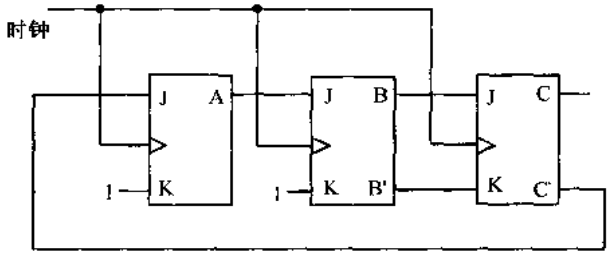


图 8-13　麦比乌斯计数器

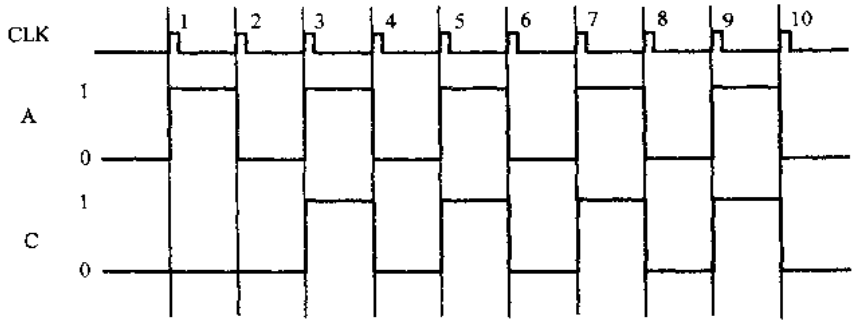


图 8-14　麦比乌斯计数器波形

8.2　对图 8-15(a)所示的计数器，假设触发器为第 7 章讨论的主-从型的器件。如果所有触发器在时钟脉冲的正跳沿触发，并假设初始化清零。画出 A 和 C 处 10 个时钟脉冲的波形(忽略传输延时)。



(a) 计数器



(b)　波形

图 8-15

解 同第 7.9 题一样，列出每个脉冲后触发器 JK 输入值的表(见表 8.3)。图 8-15(b)所示的波形可从表中读取逻辑值画出。

表 8.3

脉冲	A	B	C	$J_A(=C')$	K_A	$J_B(=A)$	K_B	$J_C(=B)$	$K_C(=B')$
0	0	0	0	1	1	0	1	0	1
1	1	0	0	1	1	1	1	0	1
2	0	1	0	1	1	0	1	1	0
3	1	0	1	0	1	1	1	0	1
4	0	1	0	1	1	0	1	1	0

无休止地重复第 3 和第 4 步

8.3 如果图 8-16 电路中的触发器初始化清零，并在时钟 1 到 0 转变时触发，D 的波形在图 8-16b 中给出，画出 F 处的宏时序图。假设 A 是多路选择器选择线的最高位。

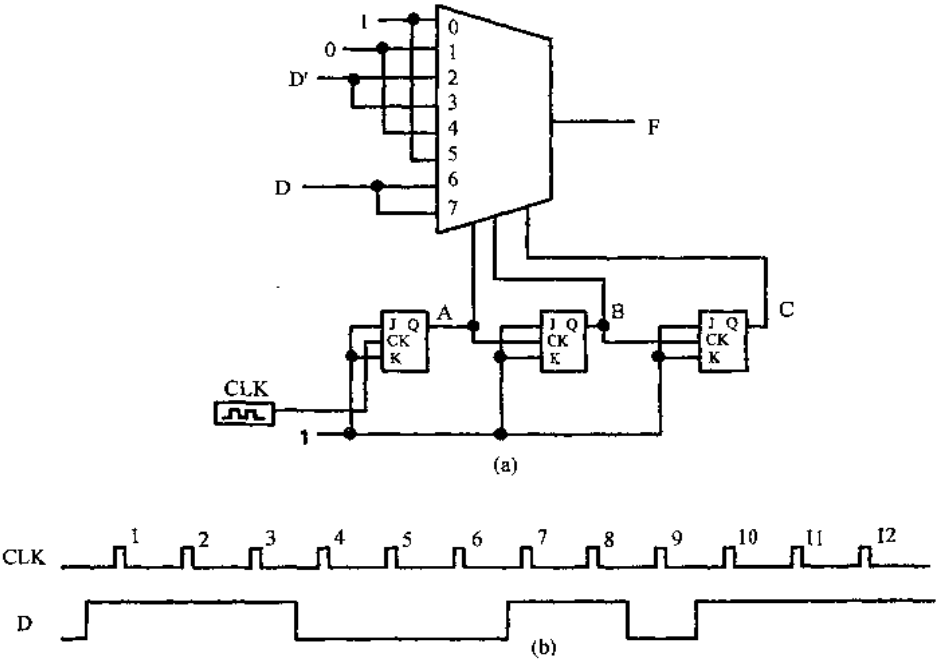


图 8-16

解、提示：触发器 1 到 0 的转变使右边的触发器翻转，因此，构建了第 8.3 节描述的波纹计数器。状态序列如表 8.4 所示，从中推导出了图 8-17 所要求的波形。

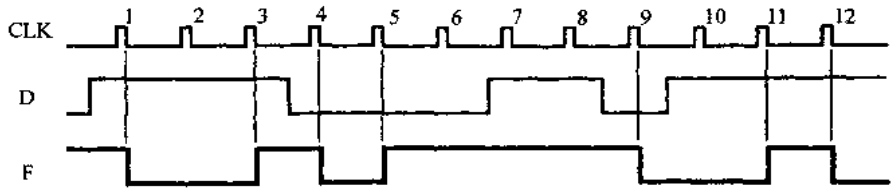


图 8-17

表 8.4

脉冲	D	选择端 A B C	多路转换器输入	多路转换器输出
1	1	1 0 0	4 (0)	0
2	1	0 1 0	2 (D′)	0
3	1	1 1 0	6 (D)	1
4	0	0 0 1	1 (0)	0
5	0	1 0 1	5 (1)	1
6	0	0 1 1	3 (D′)	1
7	1	1 1 1	7 (D)	1
8	1	0 0 0	0 (1)	1
9	0	1 0 0	4 (0)	0
10	1	0 1 0	2 (D′)	0
11	1	1 1 0	6 (D)	1
12	1	0 0 1	1 (0)	0

8.4　图 8-18 所示的电路为计数器。假设它被初始化清零，求出计数序列。（提示：与第 10 章关于状态机的内容相关。从逻辑图中，找出触发器 D 输入的方程式，

$$D_W = W'$$
$$D_X = X \oplus W + Y \oplus Z = XW' + X'W + YZ' + Y'Z$$
$$D_Y = WXY' + X'Z + W'Z$$
$$D_Z = W'Z + Y'Z' + X'Y$$

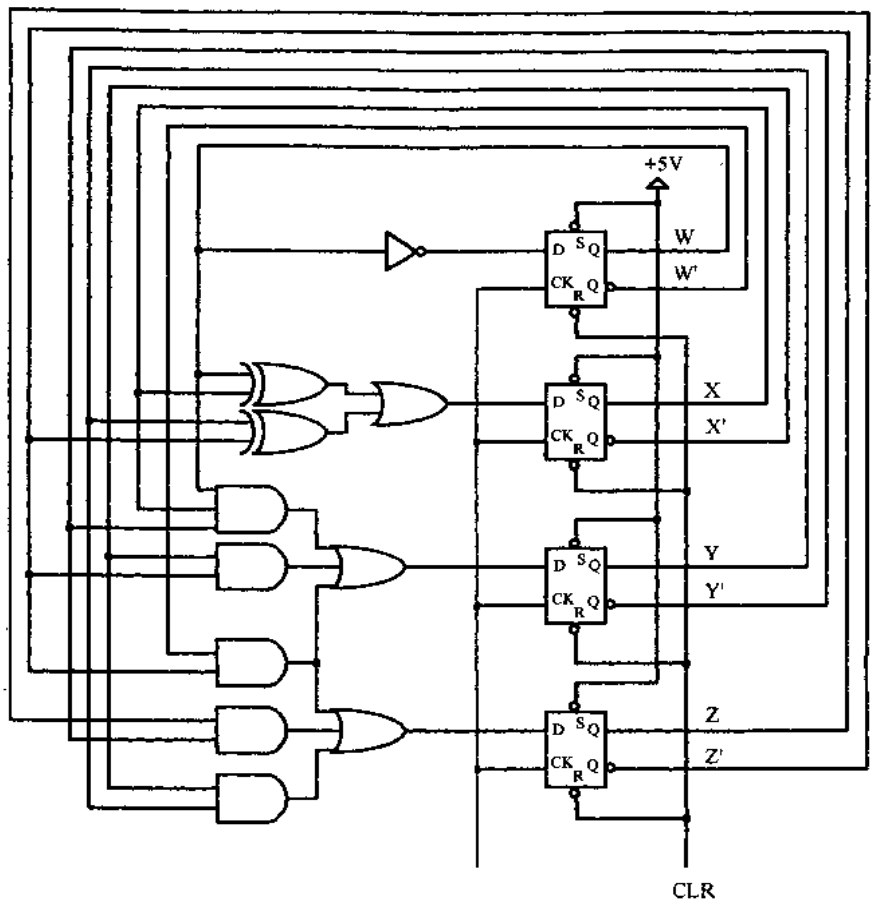


图 8-18

画出这些方程的卡诺图，产生了图 8-19。

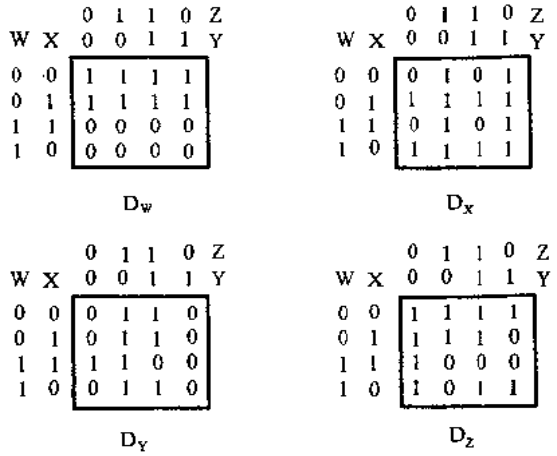


图 8-19

解 假设对于所有卡诺图可将 WXYZ 坐标看作是计数状态，而卡诺图每一项代表相应的触发器的 D 输入。初始时，计数器为 0000。这意味着从卡诺图的左上角读 4 个 D 触发器的输入，这种情况下产生了 1001。这告诉我们加上时钟之后，触发器输出会假设为组合状态 WXYZ = 1001。现在卡诺图的坐标移到了 1001(最后 1 行，第 2 列)，这里的项是 WXYZ = 0110。过程继续，产生了表 8.5

表 8.5

脉冲	W	X	Y	Z
0	0	0	0	0
1	1	0	0	1
2	0	1	1	0
3	1	1	0	0
4	0	0	1	1
5	1	0	1	1
6	0	1	1	1
7	1	1	1	1

重复

8.5 前面已经讨论过触发器问题，这在带有选通门的波纹计数器中也能出现(图 8-7)。为方便起见，重新画出图 8-20 所示的电路。

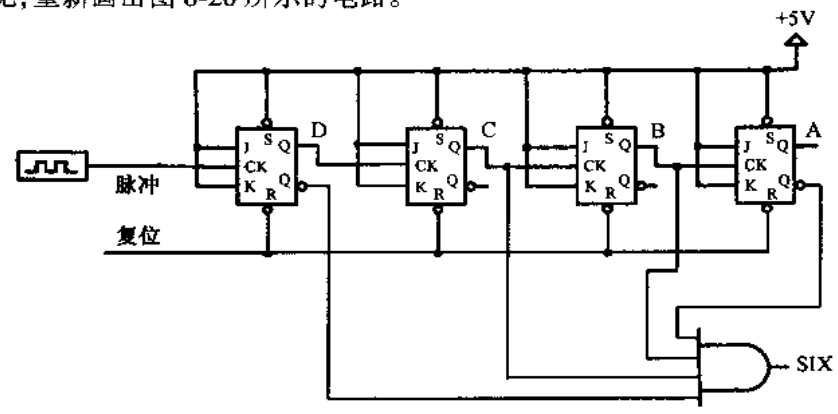


图 8-20

1到0时钟翻转使触发器翻转的情况，使得所连的对偶数响应的选通门不会遇到冲突，说明这一情况。

解 ☞ 每次最低位(D)变为1，获得偶数。因为这里0到1的跳变，不会引起触发器C跳变，也不会引起其他任何触发器的跳变。因此，不会出现波纹，也就没有冲突。

8.6 参考8.5题，对计数器状态设为0000响应的选通门，其输出端在什么情况下有可能发生冲突？

解 ☞ 考虑当波纹出现的时候。如前所示，偶数的出现不会引起波纹。因此研究奇数的出现，当然它一定紧跟着一个偶数。所以这些偶数到奇数的转变一步步地在表8.6中详细给出。0000的冲突状态由一个(X)表示出。

表8.6

1 to 2	3 to 4	5 to 6	7 to 8
A B C D	A B C D	A B C D	A B C D
0 0 0 1	0 0 1 1	0 1 0 1	0 1 1 1 (X)
0 0 0 0 (X)	0 0 1 0	0 1 0 0	0 1 1 0
0 0 1 0	0 0 0 0 (X)	0 1 1 0	0 1 0 0
	0 1 0 0		0 0 0 0
			1 0 0 0
9 to 10	11 to 12	13 to 14	15 to 0
A B C D	A B C D	A B C D	A B C D
1 0 0 1	1 0 1 1	1 1 0 1	1 1 1 1
1 0 0 0	1 0 1 0	1 1 0 0	1 1 1 0
1 0 1 0	1 0 0 0	1 1 1 0	1 1 0 0
	1 1 0 0		1 0 0 0
			0 0 0 0 所要求的门输出

8.7 图8-21所示的系统打算用作波形发生器，假设以所示序列出现清零和时钟脉冲，时钟上升沿触发，A为最高选择位，画出触发器输出端的波形(宏时序图)。

解 ☞ 在接收到CLR脉冲后，ROM地址将是000，它选择了第0条线。存储在所选地址处的数据被连在触发器的D输入端，而且指出了时钟之后的下一个状态将是什么。本例中，存储在地址000处ROM有010，这表明当下一个时钟脉冲到来时第2条线将被选择。过程继续，产生了以下的二进制序列的相应的十进制数：

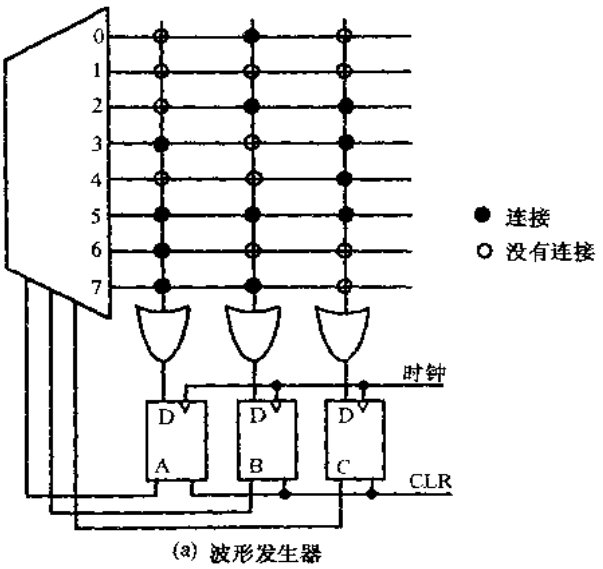
$$0\rightarrow2\rightarrow3\rightarrow5\rightarrow7\rightarrow6\rightarrow4\rightarrow1\rightarrow0。$$

所要求的波形用竖线很容易表示出来，指定0和1电平分别代表序列中的每一个状态的二进制数，如图8-22所示。

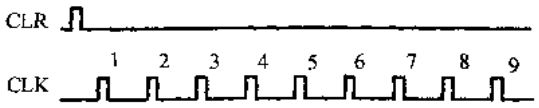
8.8 忽略选通门，考虑图8-20所示的4级二进制波纹计数器类型。假设有N级，每个触发器具有从高到低的20ns的传输延时和从低到高的10ns的传输延时(参考第7.1题)。如果每个计数必须保持50ns，求出可以计算脉冲的最大速率的表达式。假设允许计数器循环；也就是它可以从最大计数回到0。

解 ☞ 波纹式计数每一级的最长响应时间必须考虑对时钟脉冲重新出现的作用。这发生在当计数从它的最大值(2^N-1)到0时，同从表8.6中看出的一样。因为这种情况下，所有触发器经历了从1到0的跳变，使0计数出现为最后形式会花费20Nns。下一个计数将是1，而且只有第1级(LSB)会改变。因为这将是0到1的跳变，所以完成它需要10ns。一步步的分析如下

1. 计数器最初读2^N-1
2. 下一个脉冲进入计数器
3. 脉冲波纹式的通过计数器，需要20Nns完成
4. 因为计数0一定出现50ns，详细说明，在能够施加下一个脉冲前一定要经历40ns(第1级的延时为50减去10ns)



(a) 波形发生器



(b) 清零和时钟脉冲

图 8-21

脉冲 T_{min}的最小间隔是(20N + 40)ns,因此 N 级所允许的最大频率由下式给出

$$F_{max} = \frac{1}{T_{min}} = \frac{1}{20N + 40} \times 10^9 \text{ 个脉冲/秒}$$

对于 N = 4

$$F_{max} = \frac{10^9}{120} = 8.33\text{MHz}$$

参考图 8-23 可以很容易的理解这些关系。

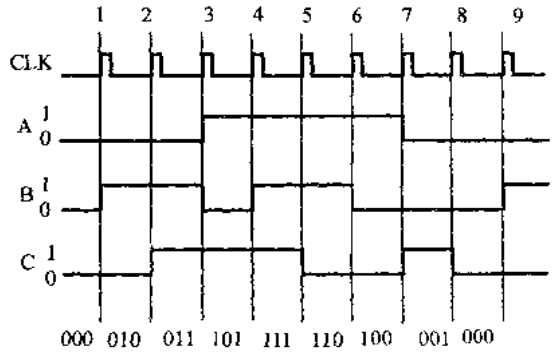


图 8-22

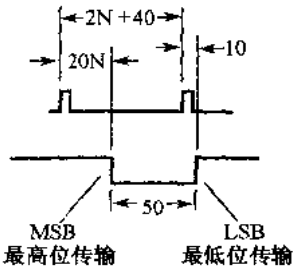


图 8-23

8.9　3 个互连的主-从 JK 触发器如图 8-24 所示。假设它们被设计成在时钟的下跳沿触发并初始化清零。

解　忽略传输延时,确定电路是否会计数,如果它计数,确定计数序列。如果电路不计数,确定在何种状态下它会挂起。

通过像第 7.9 题一样确定触发器控制信号(J 和 K)在每个脉冲前的状态,能得到答案。提示:输出 A 作为触发器 D 的时钟。结果如表 8.7 所示。

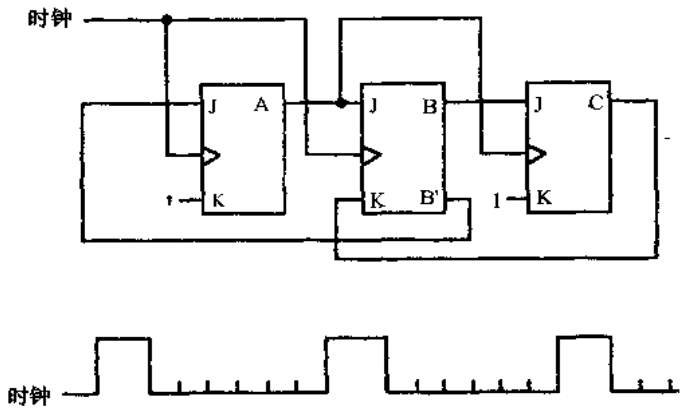


图 8-24

表 8.7

脉冲	A	B	C	$J_A(=B')$	K_A	$J_B(=A)$	$K_B(=C)$	$J_C(=B)$	K_C
0	0	0	0	1	1	0	0	0	1
1	1	0	0	1	1	1	0	0	1
2	0	1	0	0	1	0	0	1	1
3	0	1	0	0	1	0	0	1	1

因为触发器 A 的 JK 输入使 A = 0(它已是这种状态),所以在第 2 个时钟脉冲后电路挂起,触发器 B 的 JK 输入 00 表明状态零,因为 A 没有变化故而触发器 C 处没有时钟变化。

8.10 再一次假设初始化清零,画出第 8.9 题中触发器 B 和 C 输出的宏时序图。假设图 8-24 的波形中每个触发器有一个延时单元。

解 可从表 8.7 中推出波形,这里我们知道 C 保持为逻辑 0 而在第 2 个时钟脉冲处 B 输出发生跳变。跳变在时钟脉冲下降沿后一个延时间隔处发生,见图 8-25。

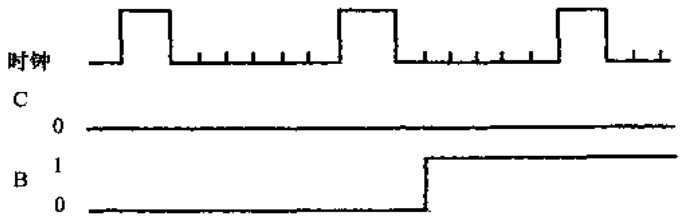


图 8-25

8.11 假设图 8-26 中的触发器初始化清零,确定计数序列(当施加时钟时,ABC 输出的连续状态)。

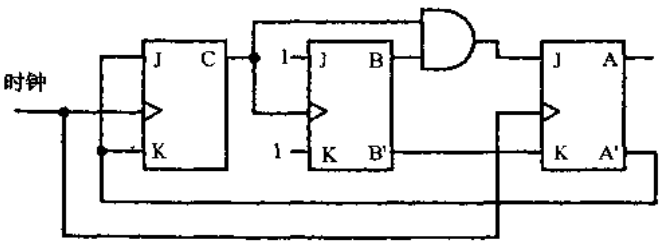


图 8-26

解 假设对时钟脉冲下降沿响应。在第 8.2 题中，列出每个时钟脉冲后 JK 输入的状态，来确定每个触发器随后的状态(见表 8.8)。因为触发器 B 由 C 触发，在 C 输出处 1 跳变到 0 之后 B 才会改变状态，而与它的 JK 输入无关。因此，一定要首先分析 C 的状态。电路从 0 到 4 重复计数。

表 8.8

脉冲	A	B	C	$J_A(=BC)$	$K_A(=B')$	J_B	K_B	$J_C(=A')$	$K_C(=A')$
0	0	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1
2	0	1	0	0	0	1	1	1	1
3	0	1	1	1	0	1	1	1	1
4	1	0	0	0	1	1	1	0	0
5	0	0	0						

有趣的是注意到，如果试着没有重新将其设置为 000 而启动计数器，有两种状态电路实质上已被锁存并不会响应时钟脉冲。第 1 个状态是 ABC = 110。这种情况下，触发器 A 和 C 的 JK = 00(没有发生改变)，并且，如果没有来自 C 的输出，B 不能翻转，所以电路保持不变。第 2 种情况是 ABC = 111。这里，触发器 A 的输入是 JK = 10，它编出了 1 个触发器已存在的状态。触发器 C 的 JK = 00，表明在 C 上不会有改变发生，因此，在 B 上也是。所以电路不会启动计数。

表 8.8 所示的正常计数过程之外的惟一剩余的状态是 ABC = 101。很容易表明如果从这个状态启动，电路会在一个时钟脉冲后 001 处进入它的正常序列。电路的序列计数特征可以如图 8-27 表示出来。

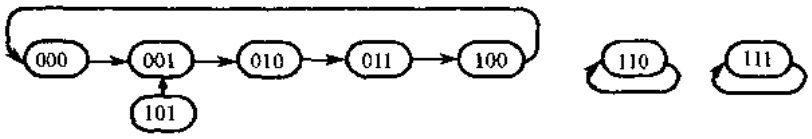


图 8-27

8.12 用于与速率乘法器输出相关的术语是异步，它指的是脉冲间隔最大和最小的比率。对图 8-8 的速率乘法器分别求出当将其比率设为 5/16 和 11/16 时的异步。

解 当比率为 5/16 时，G_2 和 G_4 工作并且 G_1 的脉冲插入到 G_4 的脉冲间。参考图 8-28，知道 G_1 的脉冲恰恰出现在一对 G_4 脉冲的中间。如果脉冲周期是 T，G_4 脉冲的间隔是 4T，G_1 脉冲的间隔是相邻 G_4 脉冲的一半，或者是 2T，因此，异步为 4T/2T = 2。

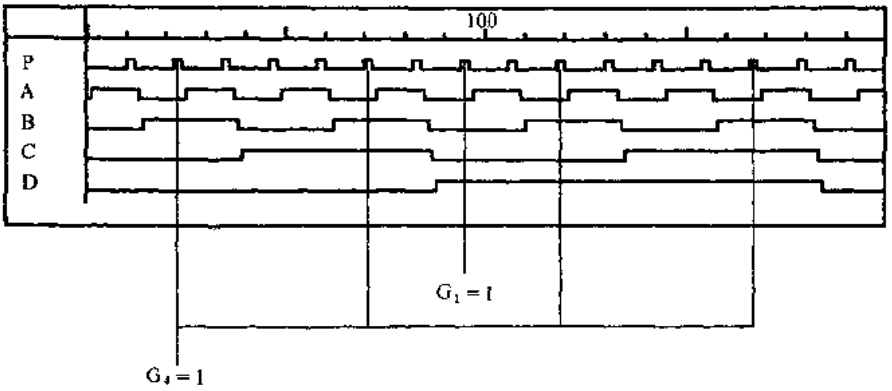


图 8-28

对于 11/16 的比率，来自 G_8 的脉冲散落在 G_2 和 G_1 脉冲之间。两个连续的 G_8 脉冲的最大间隔是 2T；最小间隔是两个连续输入脉冲的距离 T。异步也是 2。

8.13 考虑图 8-2 的锁存寄存器。假设输入线 A 和 D 上的数据以 2M Hz 的速率改变，而以

10M Hz 的速率给寄存器加时钟。用 $T_w = 0.5s$ 和 $T_R = 0.75 \times 10^9 s^{-1}$ 计算 MTBF(平均故障间隔时间)。对 T_R 指数减少 1 级的情况重新计算。

解 运用第 7.10 节的 MTBF 的方程,

$$MTBF = \frac{1}{F_{clk} \times F_{data} \times T_w \times \exp(-T_R \Delta t)}$$

这里

$$F_{clk} = 10^7 s^{-1}$$

$$F_{data} = 2 \times 10^6 s^{-1}$$

$$\Delta t = 10^{-7} s(\text{时钟周期})$$

$MTBF = 3.73 \times 10^{19} s$, 对于所有实际目的,没有亚稳特征。

在第 2 种情况下, $T_R = 0.75 \times 10^8 s^{-1}$, $MTBF = 1.8 \times 10^{-10} = 0.18ns$, 现在亚稳特征问题严重。提示:故障率的关键是依赖于技术的参数 T_R。

8.14 如图 8-29 所示的电路称作伪随机二进制序列(PRBS)发生器。在任何时钟脉冲前,施加一个负跳变的初始化脉冲。时钟开始后,说明寄存器在重复之前寄存器通过所有可能的状态(除了 0000)而改变的状态。这样一个序列的产生决定了电路用作最大长度 PRBS 发生器。

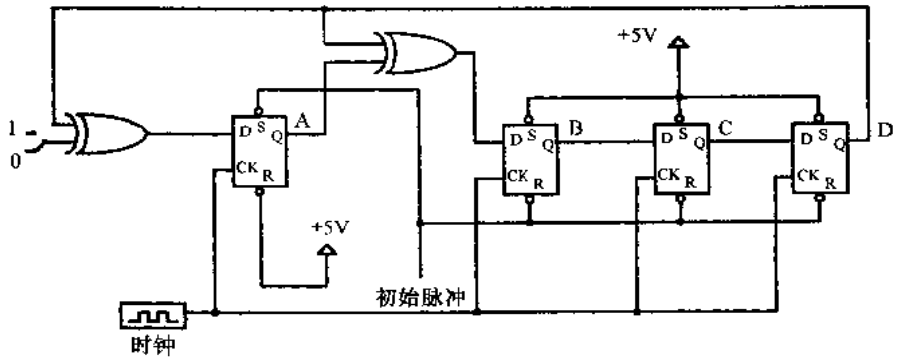


图 8-29

解 初始化脉冲设置触发器 A,复位了其余的触发器。因此初始状态是 ABCD = 1000。在每个时钟脉冲之前,考察异或门的输出,确定 D 输入值如下:

$$D_A = Q_D$$

$$D_B = Q_A \oplus Q_D$$

$$D_C = Q_B$$

$$D_D = Q_C$$

它们在每个时钟脉冲后变成了触发器的值,产生了如表 8.9 所示的序列。提示:例如第 3 个时钟脉冲之后,触发器 D 输出处的逻辑 1 使 D_A 和 D_B 的输入等于 1,并且这些值在第 4 个脉冲时传到了 A,B 的输出。第 7 个脉冲后,A 和 D 都等于逻辑 1,G_1 门有 0 输出,它表明了在第 8 个脉冲时 B 的值为 0。

表 8.9

脉冲	A	B	C	D	脉冲	A	B	C	D
0	1	0	0	0	8	1	0	1	0
1	0	1	0	0	9	0	1	0	1
2	0	0	1	0	10	1	1	1	0
3	0	0	0	1	11	0	1	1	1
4	1	1	0	0	12	1	1	1	1
5	0	1	1	0	13	1	0	1	1
6	0	0	1	1	14	1	0	0	1
7	1	1	0	1	15	1	0	0	0

因为在重复发生前经历了 15 个脉冲，所以电路是最大长度 PRBS 发生器。仿真时序如图 8-30 所示，这里人为选择 ABCD = 0100 状态来表明复位行为。

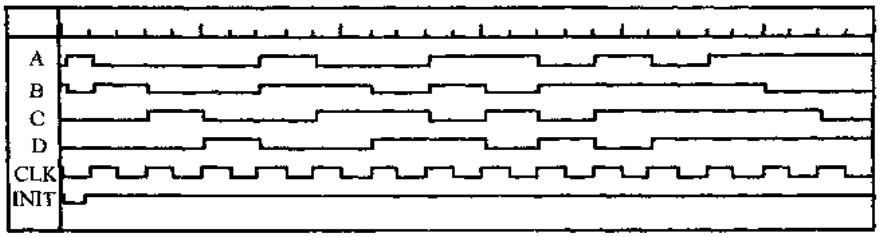


图 8-30

8.15　图 8-31 所示的电路表示 PRBS 发生器的另一种形式。可将模 2 加法器看作是对输入进行奇偶检测的电路，当进行了奇校验时它的输出产生 1。移位寄存器没有详细图示；相反，它的 5 级由方块来表示，连在与门上以及回送到加法器上的信号都来自它们，并且认为是来自它所在级触发器的 Q 输出。

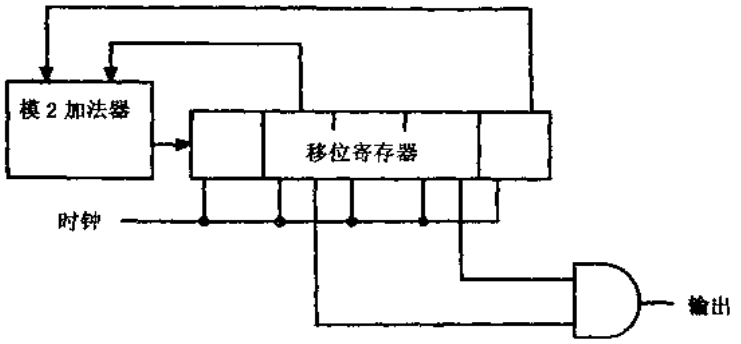


图 8-31

解　如果初始化给寄存器装入 10000，表明电路作为最长 PRBS 发生器，如第 8.14 题所定义的，并且与门产生 1 的概率等于 8/31。

将移位寄存器的级从左到右记为 ABCDE。每个脉冲后的状态如表 8.10 所示。

表 8.10

时钟	ABCDE	输出	时钟	ABCDE	输出
0	10000	0	16	11110	1
1	01000	0	17	11111	1
2	10100	0	18	01111	1
3	01010	1	19	00111	0
4	10101	0	20	10011	0
5	11010	1	21	11001	0
6	11101	0	22	01100	0
7	01110	1	23	10110	0
8	10111	0	24	01011	1
9	11011	1	25	00101	0
10	01101	0	26	10010	0
11	00110	0	27	01001	0
12	00011	0	28	00100	0
13	10001	0	29	00010	0
14	11000	0	30	00001	0
15	11100	0	31	10000	0

因为需要 31 个时钟脉冲重复序列，所以将电路看作是最长 PRBS 发生器，并且，因为通过 31 个时钟脉冲产生了 8 个 1，在输出端得到 1 的概率是 8/31。

补 充 习 题

8.16 确定图 8-8 的二进制速率乘法器对于 $C_1C_2C_4C_8 = 1001$ 的输出波形。

8.17 使 $C_1C_2C_4C_8 = 1011$ 重复第 8.16 题。

8.18 一学生决定使用选通门输出，使计数器到达计数六时，将其复位于 0000，如图 8-32 所示。所选的触发器在 1 到 0 的时钟跳变时翻转，并用低复位输入将其清零。当做好电路时，加电时电路有时像设计的那样工作，有时以随机计数开始。设计缺陷是什么？

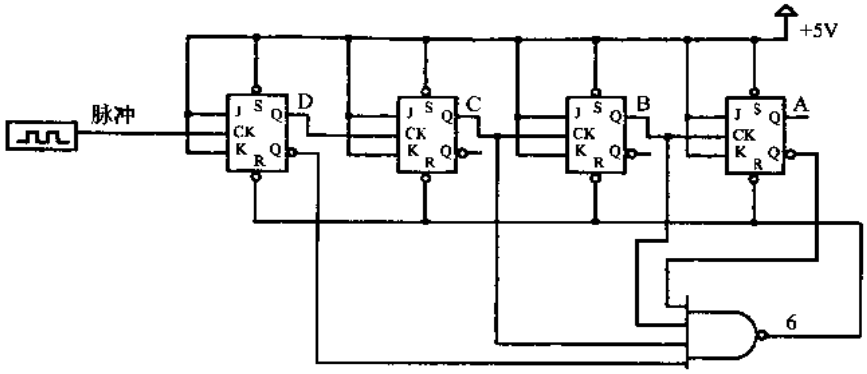


图 8-32

8.19 利用另外一个复位信号使计数器初始化为 0。用微时序图来评估你的设计。

8.20 如图 8-33 所示的电路称作单元自动装置。说明它用作最长 PRBS 发生器的原理(见 8.14 题)。

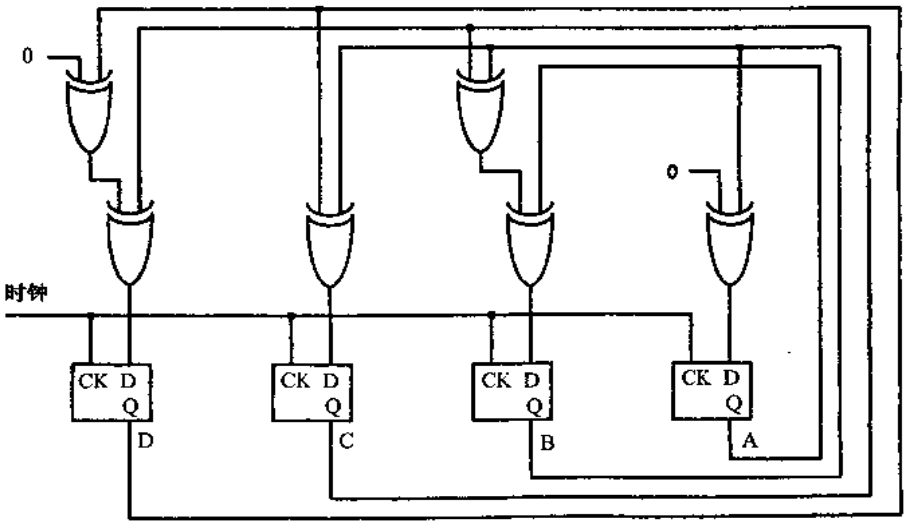


图 8-33

8.21 如图 8-34 所示的电路在第 7.8 题已研究过，那里告诉我们 T 触发器是负沿触发而 D 触发器是正沿触发。要求修改电路使得当 ABC = 011 时，操作会在这种状态冻结直到手动清零产生。用 JK 翻转触发器和 2 输入选通门来获得要求的性能。用时序图来说明工作原理。

8.22 一学生要设计从 0(0000)计数到 9(1001)，然后在第 10 个时钟脉冲时自己复位于 0000 的电路。用波形计数器和一个选通门及一个反相器结合来获得此功能。参考图 8-35，我们知道当计数到 9 时，触发器 A 和 D 在序列中第一次同时为 1，并用这些信号来启动与门，使下一个时钟脉冲加到复位线上。画

出微时序图并说明电路的工作原理。

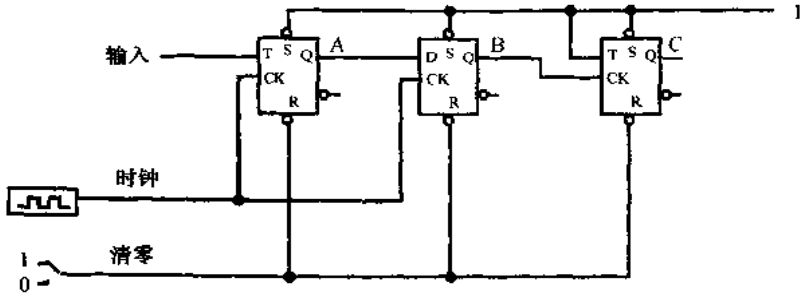


图 8-34

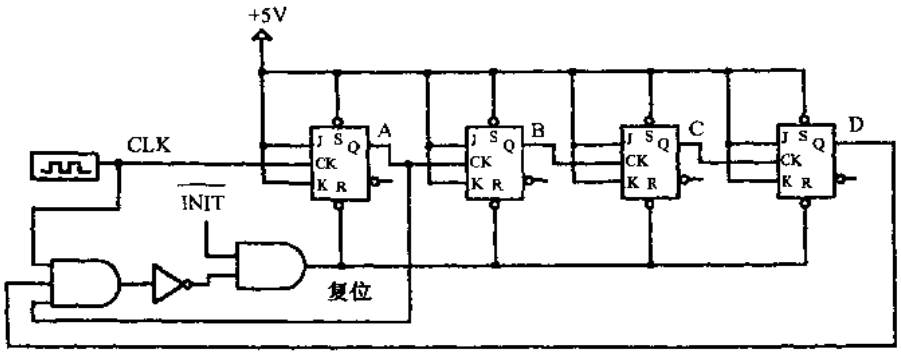


图 8-35　触发器为负沿触发

8.23　修改第 8.22 题的图，使它用作除 10 电路，其中复位线作为每 10 个时钟脉冲提供一个脉冲的输出。用时序图证明设计是正确的。

8.24　将图 8-45 的电路修改成相应的触发器，它具有置位和高电平有效而不是低电平有效的复位输入。

8.25　第 8.24 题的硬件改动对系统时序有很大影响吗？

8.26　设计 10 状态的麦比乌斯计数器，并添加选通门来识别状态 3 和 7。

8.27　对图 8-49 所示电路用正沿跳变触发器画时序图证明电路工作的正确性。假设计数器初始化清零。

8.28　设计 10 状态二进制波纹计数器，用合适的选通门来识别状态 3 和 7。

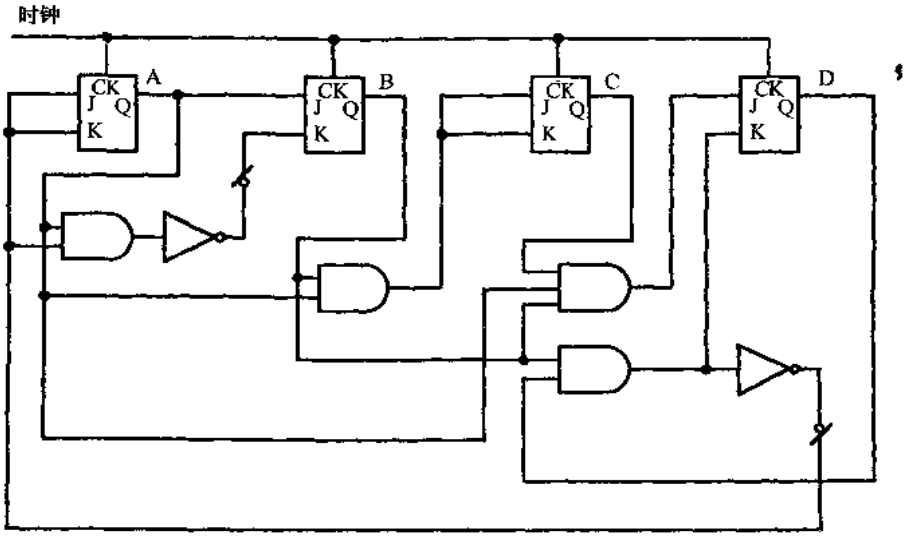


图 8-36

8.29　说明如何互连 JK 触发器链来作为 n 阶移位寄存器，当控制变量 R = 1 右移，当 R = 0 时左移。

8.30　假设图 8-36 所示的电路初始化清零为 0000，当 D 为最高位时确认计数序列。

8.31　当从二进制数 1111 开始计数时重复第 8.30 题。

8.32　到 D 触发器寄存器的输入由图 8-37 所示的逻辑驱动。假设触发器在 t = 0 时清零，确定计数序列。

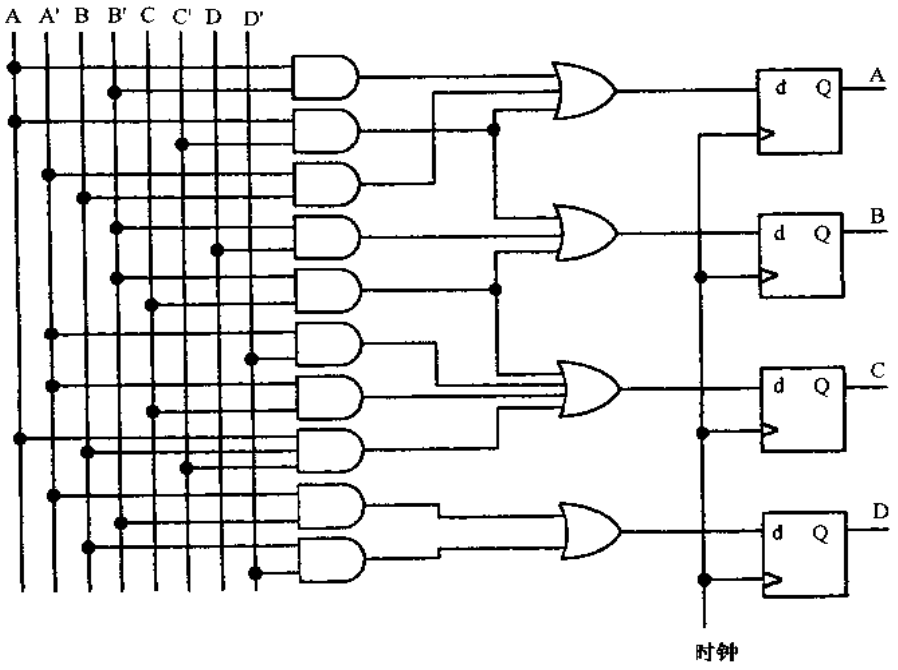


图 8-37

补充习题答案

8.16　见图 8-38(微时序图)。

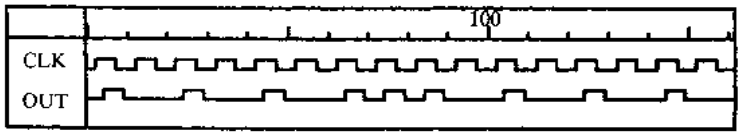


图 8-38

8.17　见图 8-39。

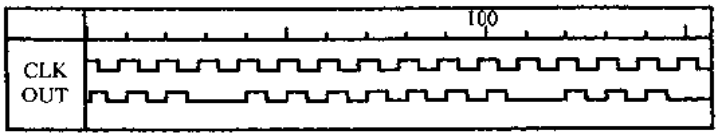


图 8-39

8.18　没有提供初始化电路，加电时，可假设从 0(0000)到 15(1111)的任何计数。

8.19　修改的电路如图 8-40 所示，其中图除了添加部分。可用选通门或初始化信号来强制复位。运用仿真可以得到微时序图(见图 8-41)。提示：状态 6 仅存在到这个值被识别，传输经过与门和异或门，然后使波纹计数器复位。这可能是严重的问题，特别是在高时钟频率时，设计者必须考虑到这一点。

8.20　当为非零起始点时，假设为 ABCD = 1000。如表 8.11 所示时钟序列在重复发生前用了 15 步，因此这是 4 触发器的最大长度。

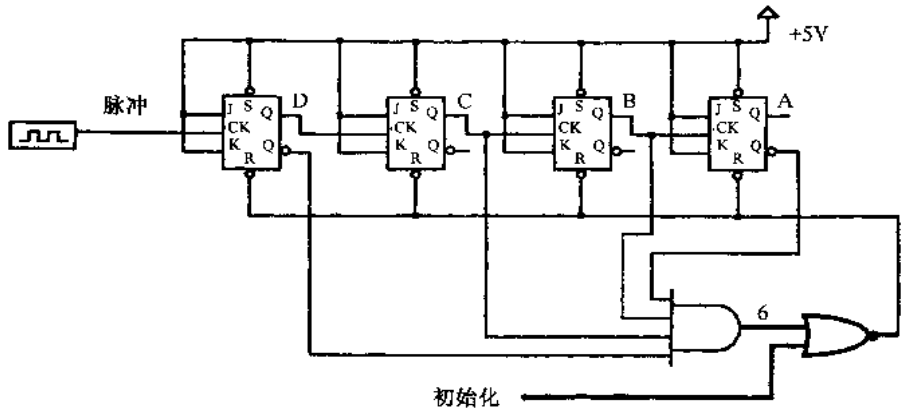


图 8-40

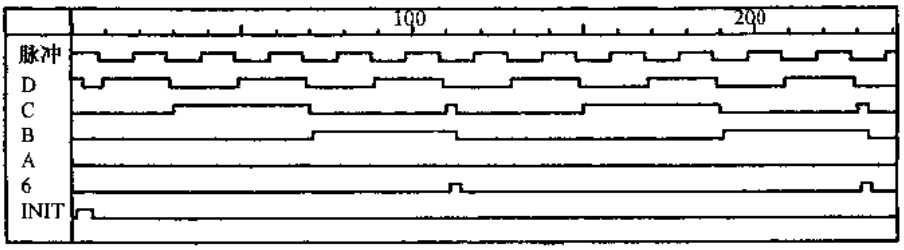


图 8-41

表 8.11

ABCD	ABCD	ABCD	ABCD
1000	1100	0110	1101
0100	1010	1011	1001
1110	0001	0010	0111
1111	0011	0101	1000

重复

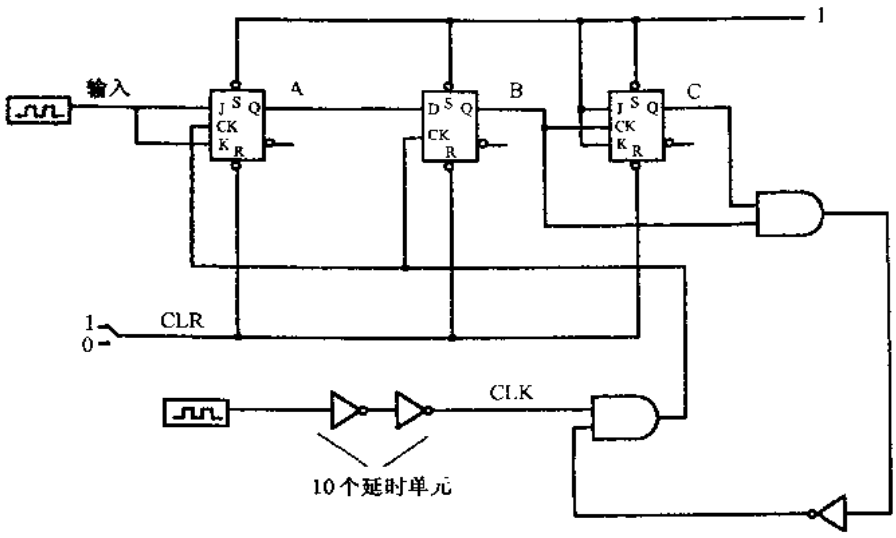


图 8-42

8.21　见图 8-42 和图 8-43。关于这个问题的仿真讨论可在附录 C 中找到。

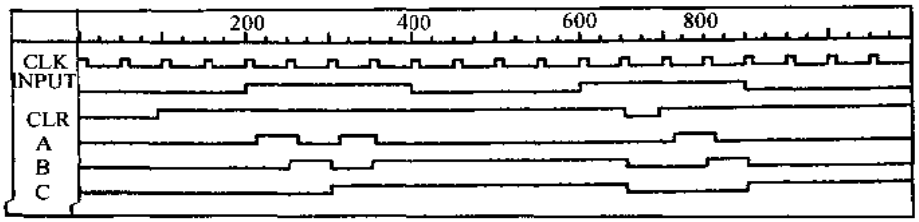


图 8-43

8.22　参考图 8-44。激活复位与时钟信号同样也推进计数到 1。越过了 0 计数。因为每 9 个时钟脉冲产生 1 个复位脉冲，所以计数器成了极好的除 9 电路。

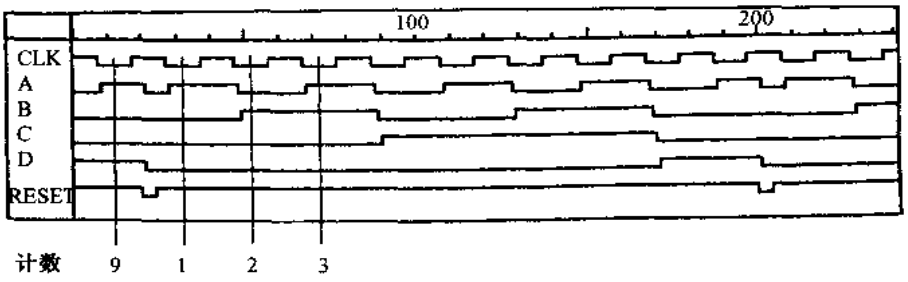


图 8-44

8.23　见图 8-45 和图 8-46。

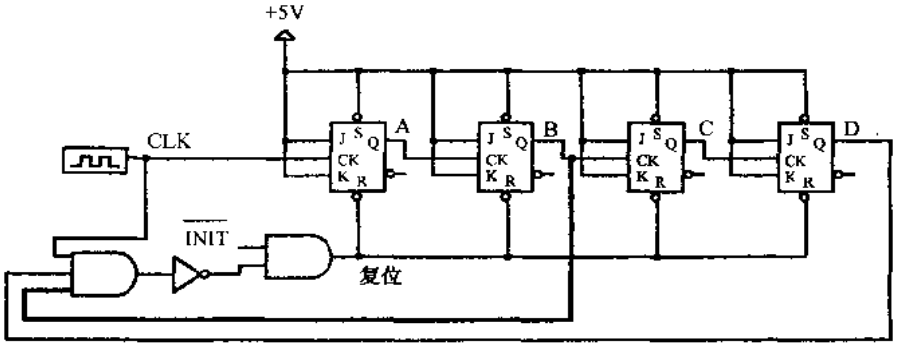


图 8-45

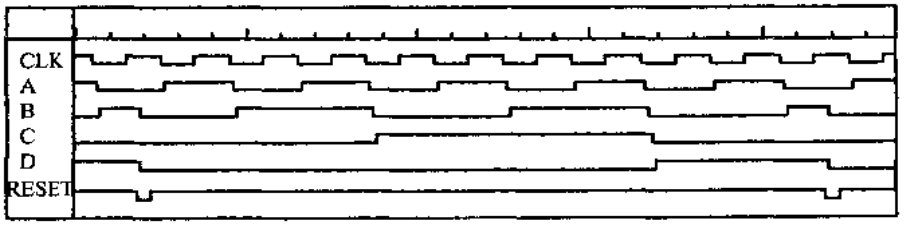


图 8-46

8.24　见图 8-47，参考第 4.39 题。

8.25　比较图 8-46 和图 8-48 的微时序图，我们看到了这两种情况下，电路在复位到 0 前保持在计数 10 的时间比半个时钟周期要长一些。第 2 种情况下，复位脉冲要窄 1 个传输延时间隔，如果时钟频率极大的

提高，可能很快会出现亚稳问题。

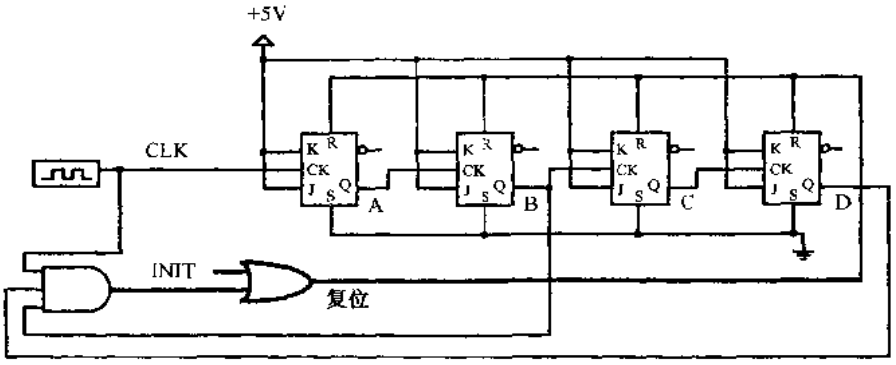


图 8-47

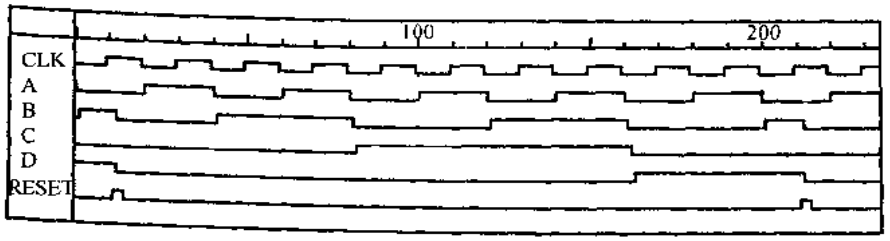


图 8-48

8.26　见图 8-49。提示：2 输入门足够了。

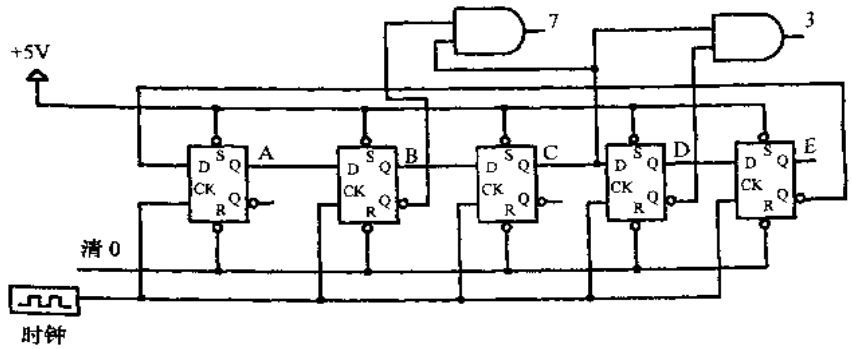


图 8-49

8.27　见图 8-50。

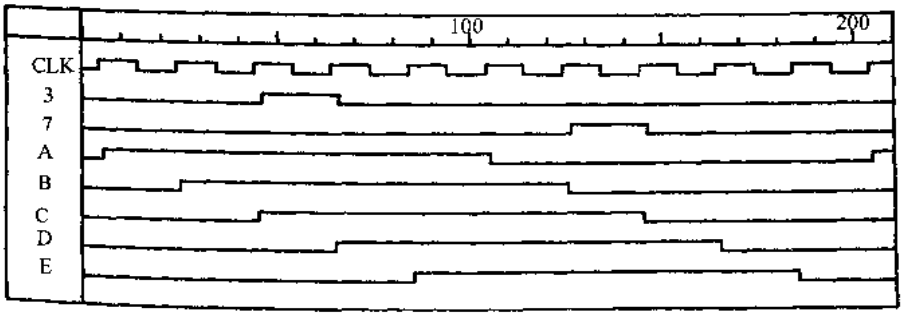


图 8-50

8.28　见图 8-51。

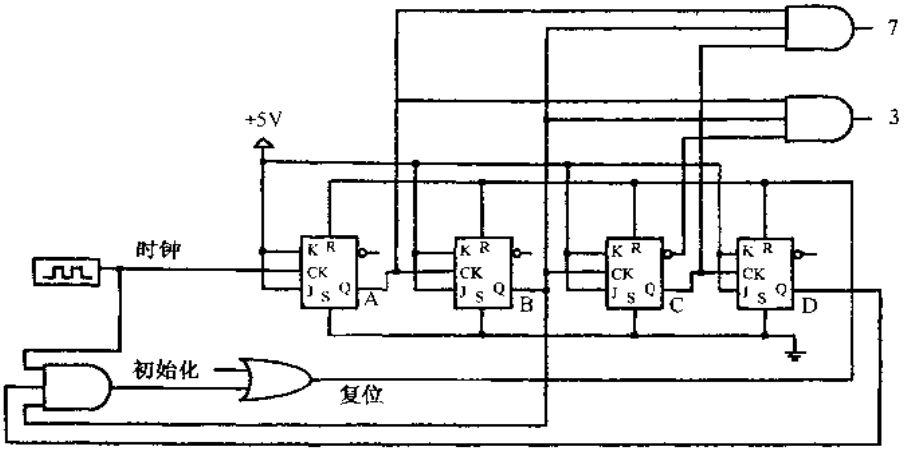


图 8-51

8.29　激励方程是 $J_n = RQ_{n-1} + R'Q_{n-1}$ 和 $Kn = RQ'_{n-1} + R'Q_{n-1}$。这个逻辑电路的实现表示在图 8-52。应与图 8-4 进行比较。

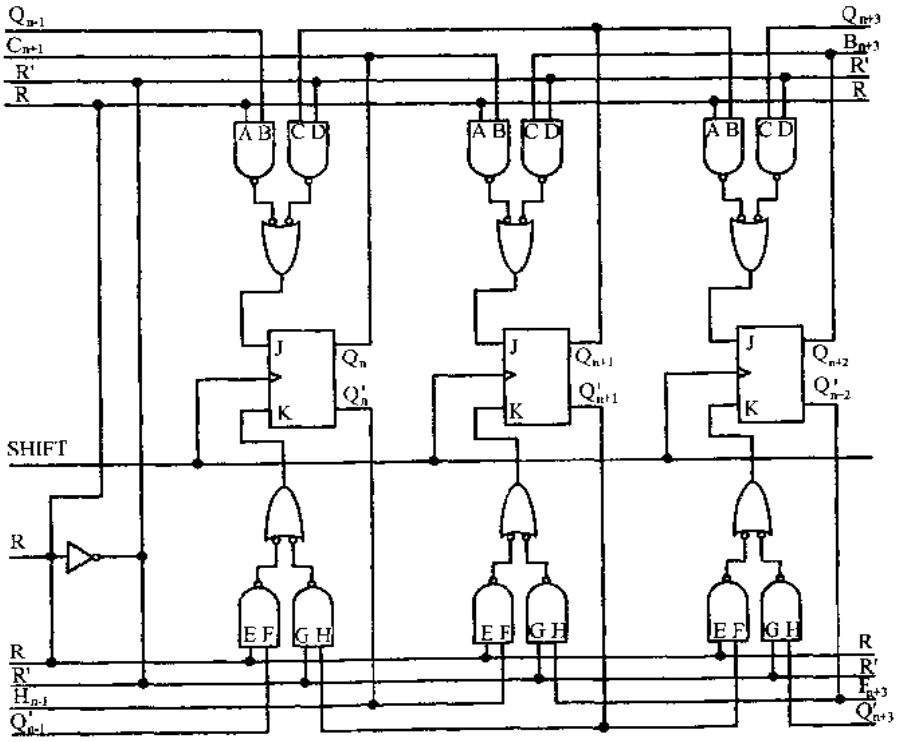


图 8-52

8.30　从 0 开始，十进制计数交替进行 1.2，1，2，…

8.31　十进制计数是 15，1，2，1，2，…

8.32　十进制计数是 0，3，7，10，14，1，5，8，12，15，0(重复)。

第9章 专用器件

9.1 引言

市场上很多数字逻辑的器件应用在很大程度上不能满足各种客户需要，因此，专用 MSI 和 LSI 芯片的开发市场非常大。这种情况导致可编程逻辑器件(programmable logic device-PLDs)的开发，即能够制造出高密度并容易满足不同用户专门应用的器件。

programmable logic device (PLD)分成两大类，第 1 类为专门应用而进行程序设计的电路引起芯片内物理变化，这将在本章讨论。第 2 类是逻辑功能被电信号控制的器件，将在第11章讨论。

面对客户的 IC 市场，另一种方法是创造一种能够容易在硅片上制造的标准元件库。此元件库为计算机-可读形式，包括制造元件需要的光刻信息方法和内部互连的设计规则。强大的计算机硬件和软件为用户组合产生客户化硅片。这种方法得到的电路被称为客户 IC，或更通俗地称为专用 IC 集成电路(application-specific integrated circuits-ASIC)。

注意：上面提到 PLD 也是专用器件。各种类型之间的区别通常建立在这样的基础之上，即生产商提供的结构类型，程序设计技术的特性要求，擦除程序的能力和所要求的时间等等。

9.2 程序设计技术

各种技术被用于实现可编程逻辑。

可熔丝连接

在这种情况下，逻辑是由所有可能的用户——决定内部连接的构成形式。并且，这种连接早在工厂中就预先做好了。这些连接用熔丝材料制造，而器件被做成专用的，通过烧断熔丝去掉连接。芯片设计的电流相对比较大，在程序设计工作期间，专门的可熔丝连接由用户选择。程序设计的方法显然是不可逆的。

紫外线可擦除程序设计

这种器件通过所有可能的用户，选择连接点或不连接点而构成。这些区域通过使用适当的电压在程序设计期间有选择性地激活每个期望被访问的连接点。电压产生稳定的“电荷包”在硅片上建立起连接。在紫外线可擦器件中，IC 封装上有一个小窗口，位于芯片上面，允许芯片暴露在紫外线下面，电荷包形成的程序连接允许被解除，即基本上“熔掉”他们。暴露在适当强度的紫外线下 10 分钟左右就会完全被擦除，器件可以重新编程。

电可擦除程序设计

有很多应用要求擦除时间较快，甚至有必要小于 10 分钟。这种情况要求完成电擦除，基本上是可逆的程序设计过程。这种类型的某些器件中，可以选择性地擦除一部分，而其他器件必须整个芯片被清除。高速电擦除又称“闪电”擦除，能在几秒中内完成。

就用户关心的问题来说，可擦除和可熔丝连接的器件编程过程基本相同。

9.3 PROMS 和 EPROMS

可编程只读存储器(PROM)是有特殊结构的可编程器件。它是由用户-可控连接成或门输入的 ROM(见 5.4 节)组成的。这些器件是可熔丝连接型式(PROM)，紫外线可擦除型式(EPROM)和电可擦除型式(EEPROM)。所有 PROM 都包含全译码输出结构，以致使每一个

输出与任何或门输入之间的连接是可能的。其结构如图 9-1 所示。

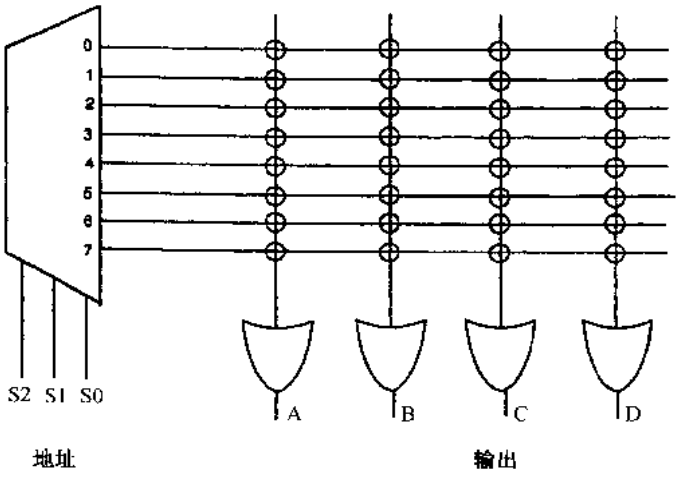


图 9-1　PROM 结构 (N=3)

有 N 个地址位的全译码器产生 2^N 个输出线的任何一个信号，考虑真值表中的每一行。由于输入的逻辑信号起地址位作用，而译码器对 N 个地址位有 2^N 条输出线。每次一个新的逻辑输入被引入，由于输入的逻辑信号起地址位作用，而译码器对 N 个地址有 2^N 条输出线。每增加一个新的逻辑输入变量，译码器输出数加倍。PROM 对于实现多个逻辑输入不是特别有用，因为译码器需要大的硅片面积。然而，其结构不允许设计者列用一些"无关最小项"的条件。ROM, PROM, EPROM 等，主要应用是给计算机应用存储命令，而不是实现逻辑。

PROM 或 EPROM 编程要求某些计算机软件和"PROM 编程器"附加在计算机编程口上或是作为计算机的一个终端。一般用二进制数表示或门连接状态(1 表示连接，0 代表断开)。PROM 或 EPROM 被装入的程序是以十六进制数表示连接形式和地址的。数据是经键盘输入，并经软件把它转换成为一组电信号，计算机通过 PROM 编程器把程序烧成所期望的连接型式。

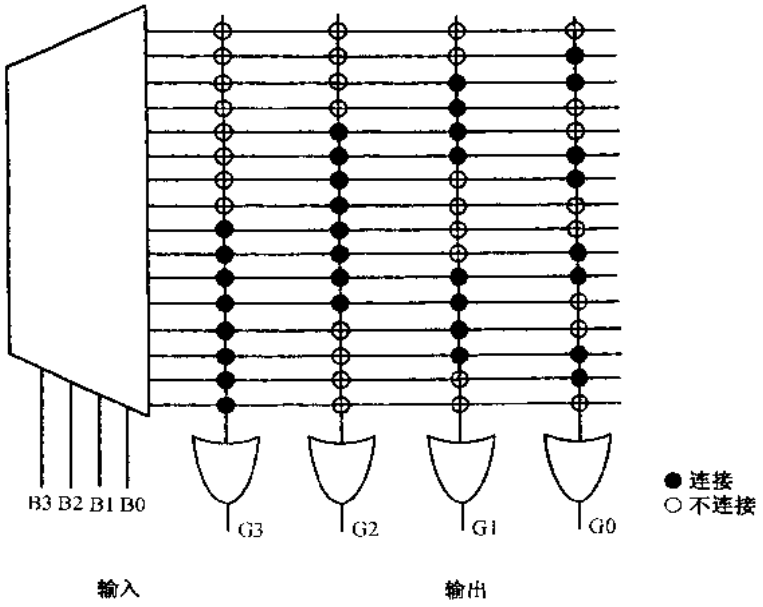


图 9-2　将二进制数转换成格码的已编程 PROM

例 9.1 对 EPROM 编程，将 4 位二进制数转换成等效的格雷码。这个问题的反问题和相应的真值表如问题 5.8 所示。

把二进制数看成指定地址的查找表，对应的内容为格雷码数。如图 9-2 所示，实际程序起始地址为 0000，而十六进制数列成表为 0, 1, 3, 2, 6, 7, 5, 4, C, D, F, E, A, B, 9, 8。

9.4 可编程阵列逻辑(PAL)

PAL(programmable Array Logic) 是一种特别有用的可编程器件，相应的地址输入位数很多，而编程数量受到限制，这些被约束的函数产生无关最小项，这意味着真值表中很多行不需要实现。

像在 ROM 中那样，基本结构是与门驱动或门。在 PAL IC 中，预先连好的或门输入送到有限数量的与门，留给设计者自由地选择与门输入。这种限制常常是可以接受的，因为无关最小项的存在，只要求有限数量的或门连接，对逻辑的输入可以是直接形式也可是逻辑反形式，逻辑又是在器件内部产生的。

或门被连接到具有用户可选输入的与门的子集上。

有些 PAL 器件只由组合元件(与门、或门和反相门)组成，而其他一些也有包含几个触发器在里面，以至子完整的小型状态机可以放在单个芯片内。普通型 PAL 结构，包含有触发器的如图 9-3 所示。注意：某些与门输入直接来自芯片外部输入端，而其他来自内部触发器反馈。或门的输出或者直通到输出端(无寄存器型)或者通过触发器锁存再输出(有寄存器型)。设计者希望选择什么类型的器件，可通过选择相应的与门使用与否来决定。来自触发器反馈的信号，通常有直通的或是反相的两种形式。

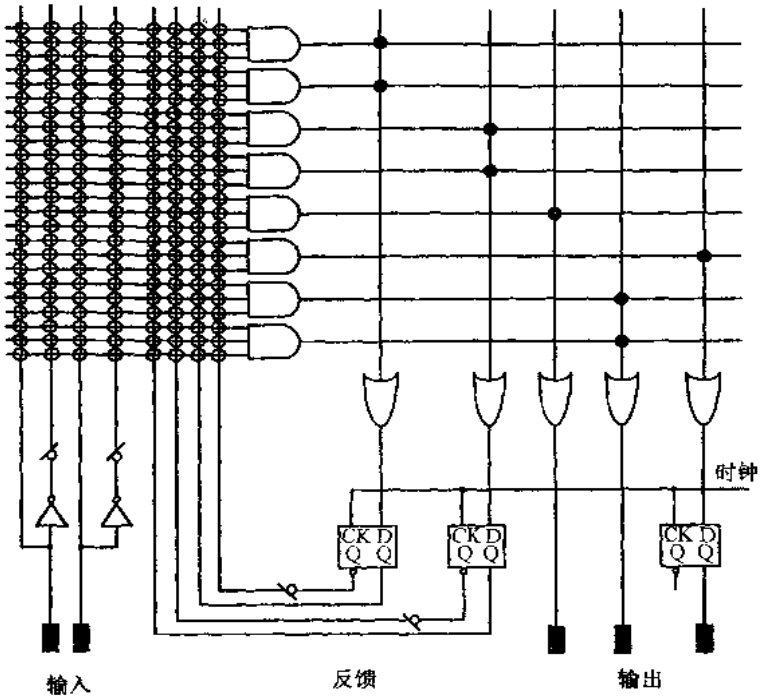


图 9-3 PAL 的一般结构

图 9-4 所示是商业化产品(PAL16R 6, 由 Advanced Micro Device, Inc. 生产)的内部结构，它包含有 8 个或门，每个或门输出由 64 个与门中 8 组之一来确定。

每个与门有 32 个输入由用户可选择连接，有些或门的输出是被寄存的而另外一些则无寄存特性。然而它们都有反馈到与门输入的能力，如上述器件中有 6 个触发器。

16R6 芯片使用了三态门缓冲元件(见图 8-12)，它起开关作用，通过使能信号控制、有选择

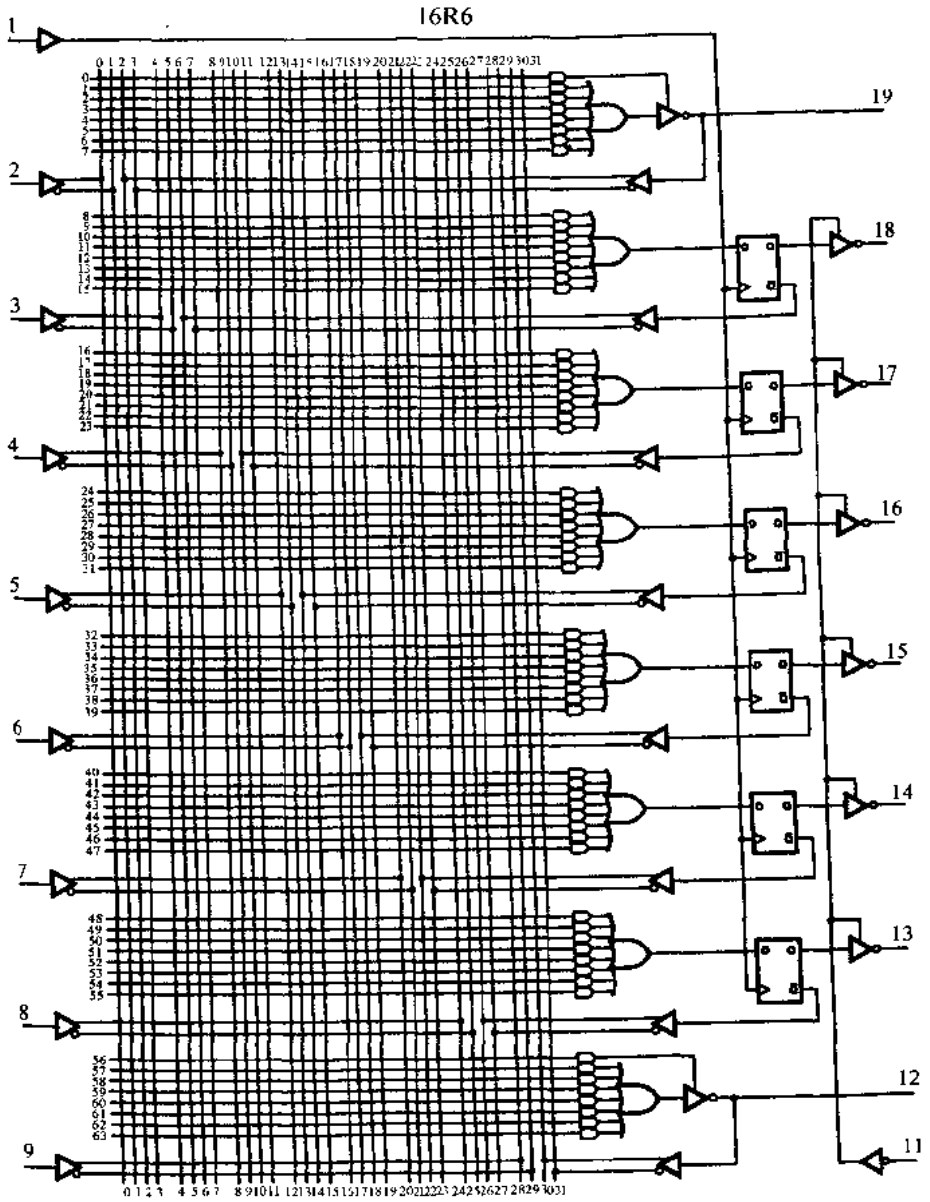


图 9-4　PAL16R6 逻辑图

地接通或断开(隔断)与它相关的输出端。

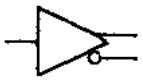


图 9-5　放大器对

图 9-5 所示是用于 16R6 逻辑图中的另一种符号，是一种简化符号，意味着单输入两输出的放大器，其中一个输出是反相(有圈表示)而另外一个则为同相。

PAL 器件编程

PAL 器件编程采用先进的计算机辅助软件和附加于计算机上的 PAL 程序写入器或终端。PALIC 的内部结构，是用与门驱动或门，给予编程信息的表达式是由真值表得到的基本布尔方程。对于 PAL 可编程器件来说，有很多专门的软件语言可得到，最早的一种叫 PALASM 在正逻辑符号的基础上，使用布尔输入非常有效。在这种语言中逻辑操作符被修改，以便使用普通键盘。按先后顺序给出一个操作符简表(不完全)如下：

；　后跟说明

/　逻辑反或有源低

· 逻辑与
+ 逻辑或
:+: 异或
:*: 异或非
= 相等
:= 寄存器输出方程运算符

如果希望输入布尔方程，F= AB′ + CD + (B + C′)′，用 PALASM 软件首先要求使用狄·摩根定理转换方程成为与-或(积之和)形式。应用上面定义的语法得：

F = AB′ + CD + B′C　　与-或
F = A*/B + C*D +/B*C　　PALASM 软件语法

如果输出 F 被寄存，等号将优先被克隆。

输出端真值考虑

观察图 9-4 PAL16R6 电路图，所示逻辑是积之和的形式，是由与门网络连接到或门，产生的输出驱动相应触发器的 D 输入端。到达的时钟脉冲依次将这些值传送到输出端，经反相放大器连接到芯片封装的引脚。这样，每个输出引脚的值与相关附属于触发器输出端相反，表示此低电平有效，我们定义 Q 为高电平有效，而触发器的 D 输入端必须用积之和的 1 传输。另一方面，如下面的例 9.2 所示，如果输出端表示成高有效，那么，这只能用附加在逻辑非上的圆圈来平衡，意味着 Q 必须由来自反相的逻辑方程驱动，或等效成包含 0。

逻辑设计过程产生的布尔方程常常对应包含 1 的积之和(见第 10 章的例子)，如果设计者接受输出形式为低有效(即引脚设计成/X)，则布尔方程可以使用本节前面的关系表直接转换成PALASM 形式。如果输出要求逻辑高有效，它必须使用逻辑非布尔表达式。很多 PALASM 软件形式，在斜线前简单地加一圆括号以括上方程(/)。观察例 9.2，本书涉及的 PAL16R 6 或 PAL16R 8 的问题均假设指定输出端低有效，以便使设计的布尔方程能直接转换成 PALASM 语法。

注意：不是所有的 PAL 器件都经反相放大器输出，所以，设计方程有很多其他方法。在 PAL 器件中，用 PALASM 软件的各种形式都可以实现。读者适当的参考详细的设计手册。

例 9.2 设计一个简单的二进制位加法器在 PAL16R8 中用寄存器输出。

解 真值表、初始化描述在第 2.4 节，如表 9.1 所示，PALASM 形式的和及进位输出变量方程是：

$$F = /X^*/Y^*/C_i + /X^*Y^*/C_i + X^*/Y^*/C_i + X^*Y^*C_i$$

$$C_o = /X^*Y^*C_i + X^*/Y^*C_i + X^*Y^*/C_i + X^*Y^*C_i$$

这里，和的输出为 F，进位输出为 C_o。

表 9.1

输入			输出	
X	Y	C_i	F	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

遵循基本的语法规则,设计者使用任何方便的文本编辑器键入 PALASM 程序的源文件。这个文件定义变量、指定布尔关系和如图 9-6 所示排列引脚号。任何以分号开始的行都会被软件忽略,其功能是解释和说明。前几行是大文件输入,接着以 PINS 开始的那些行指明了芯片的引脚号的排列必须按顺序输入。所有引脚都必须考虑到,并且 NC 用来表示缺省连接,使用 PALASM 语法下一步是布尔方程输入。

输入文本的仿真部分提供软件指令来模拟电路操作和对布尔指标进行检查。命令 SETF 表示对程序而言,跟在关键字后列出的输入变量将被应用。模拟开始首先进入 SETF/OE 表示 LT 输入被应用到 TRI-STATE 缓冲器上,因而使能输出(输出使能)。CLOCKF 命令表示时钟脉冲被加到后面指定的引脚上,在本例中是标为 CLK 的单引脚。模拟每次进行 1 步,即增加 1 个时钟脉冲就被用于跟随每套对应于真值表输入变换的输入。

当程序运行时,成功的模拟将用"无错"、"无警告"、"文件运行成功"词语来表示。

模拟结果被存在 ADDER.HST 文件名下,如图 9-7 所示。每一竖行表示时间间隔(像微定时图)。字母"c"表示 CLKF 命令,"g"表示 SETF 命令,在 3 步时钟定时作用过程中:第 1 步时钟引脚电压上升,接着记录新输出端值,最后,时钟端电压下降。每一行包括着结果来自模拟命令的引脚值高:(H),低(L),或不确定(X)。

PALASM 使用二进制位加法器 PDS 文件例

```
TITLE          ADDER
PATTERN        A
REVISION       1
AUTHOR         JIM SWATCH
COMPANY        RIT
DATE           MAY 1992

CHIP           ADDER   PAL16R8

;PINS          1       2       3       4       5       6       7
               CLK     X       Y       CI      NC      NC      NC

;PINS          8       9       10      11      12      13      14
               NC      NC      GND     OE      NC      NC      NC

;PINS          15      16      17      18      19      20
               NC      NC      NC      F       CO      VCC

EQUATIONS

     /F:=/X*/Y*/CI+/X*Y*CI+X*/Y*CI+X*Y*/CI      ;SINCE THE OUTPUTS FROM
                                                ;THE PAL16R8 ARE INVERED,
     /CO:=/X*/Y*/CI+/X*/Y*CI+/X*Y*/CI+X*/Y*/CI  ;WRITE THE EQUATIONS FOR
                                                ;THE ZERO TERMS NOT THE ONES.
SIMULATION

     SETF /OE                                   ;THIS ENABLES THE OUTPUTS
     CLOCKF CLK                                 ;GENERATES A CLOCK PULSE
     SETF /X /Y /CI                             ;THIS FIRST LINE OF THE
     CLOCKF CLK                                 ;TRUTH TABLE
     SETF /X /Y  CI
     CLOCKF CLK
     SETF /X  Y /CI
     CLOCKF CLK
     SETF /X  Y  CI
     CLOCKF CLK
     SETF X  /Y /CI
     CLOCKF CLK
     SETF X  /Y  CI
     CLOCKF CLK
     SETF X   Y /CI
     CLOCKF CLK
     SETF X   Y  CI
     CLOCKF CLK
```

图 9-6

存储在文件中的熔丝型式叫 ADDER-ex.XPT。如图 9-8 所示。这里的水平指针从 0 到 31 和图 9.4 所示电路相关的列数对应,垂直指针从 0 到 63,每行对应电路图中的一个与门,熔断的熔丝(不连接)用(—)表示,连接的用(X)表示。未使用的门,所有熔丝均不熔断,在本例中,与门 0 到 3 和 8 到 11 被用于逻辑,而其他与门保持在 0。实际的电路连接可参照指针和电路图 9-4。

```
PALASM89  PLDSIM   - DEVELOPEMENT VERSION (26-SEP-1989)
 (C) - COPYRIGHT ADVANCED MICRO DEVICES INC., 1989

PALASM SIMULATION HISTORY LISTING

Title      : ADDER                          Author    : JIM SWATCH
Pattern    : A                              Company   : RIT
Revision   : 1                              Date      : MAY 1992

PAL16R8
Page : 1
     g  cg  cg  cg  cg  cg  cg  cg  cg  c
 CLK XHHLLHHLLHHLLHHLLHHLLHHLLHHLLHHLLHHL
 X   XXXXLLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHH
 Y   XXXXLLLLLLLLHHHHHHHHLLLLLLLLHHHHHHHH
 CI  XXXXLLLLHHHHLLLLHHHHLLLLHHHHLLLLHHHH
 GND LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
 OE  LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
 F   XXXXXXLLLLHHHHHHHHLLLLHHHHLLLLLLLLHH
 CO  XXXXXXLLLLLLLLLLLLHHHHLLLLHHHHHHHHHH
 VCC HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

图 9-7

9.5 可编程逻辑阵列(PLA)

PLA 可编程逻辑器件为设计者提供了更大的自由度,因为与门阵列和或门阵的连接均可编程。它结合了 PAL IC 的与门可编程和 PROM 的或门可编程的特性。编程的逻辑阵列常常包含有寄存输出的触发器与/或反馈变量,简化的通用逻辑图为图 9-9。

像 PAL 器件一样,PLA 器件包含着直接的或互补的两种输入数据形式。控制连接到或门输入的能力为设计者提供了额外的自由。例如,同一个与门可驱动几个或门,而在 PAL 器件中,对或门中的每一个必须用双倍的与门。

PLA 的编程是在计算机或工作站上以类似 PAL IC 的方式编程,并且有关于用户-友好的软件有相当数量是可以得到的。程序以布尔形式输入,像例 9.2 说明的那样,或用状态表输入如第 10 章描述的那样。在超大规模 IC 中 PLA 常常用作基本的组件模块。

9.6 门阵列

```
PALASM89  PAL ASSEMBLER   - DEVELOPEMENT VERSION (28-AUG-1989)
 (C) - COPYRIGHT ADVANCED MICRO DEVICES INC., 1989

TITLE   :ADDER                    AUTHOR :JIM SWATCH
PATTERN :A                        COMPANY:RIT
REVISION:1                        DATE   :MAY 1992

PAL16R8
ADDER
                      11  1111  1111  2222  2222  2233
     0123  4567  8901  2345  6789  0123  4567  8901

0    -X--  -X--  -X--  ----  ----  ----  ----  ----
1    -X--  -X--  X---  ----  ----  ----  ----  ----
2    -X--  X---  -X--  ----  ----  ----  ----  ----
3    X---  -X--  -X--  ----  ----  ----  ----  ----
4    XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
5    XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
6    XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
7    XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX

8    -X--  -X--  -X--  ----  ----  ----  ----  ----
9    -X--  X---  X---  ----  ----  ----  ----  ----
10   X---  -X--  X---  ----  ----  ----  ----  ----
11   X---  X---  -X--  ----  ----  ----  ----  ----
12   XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
13   XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
14   XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
15   XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX  XXXX
```

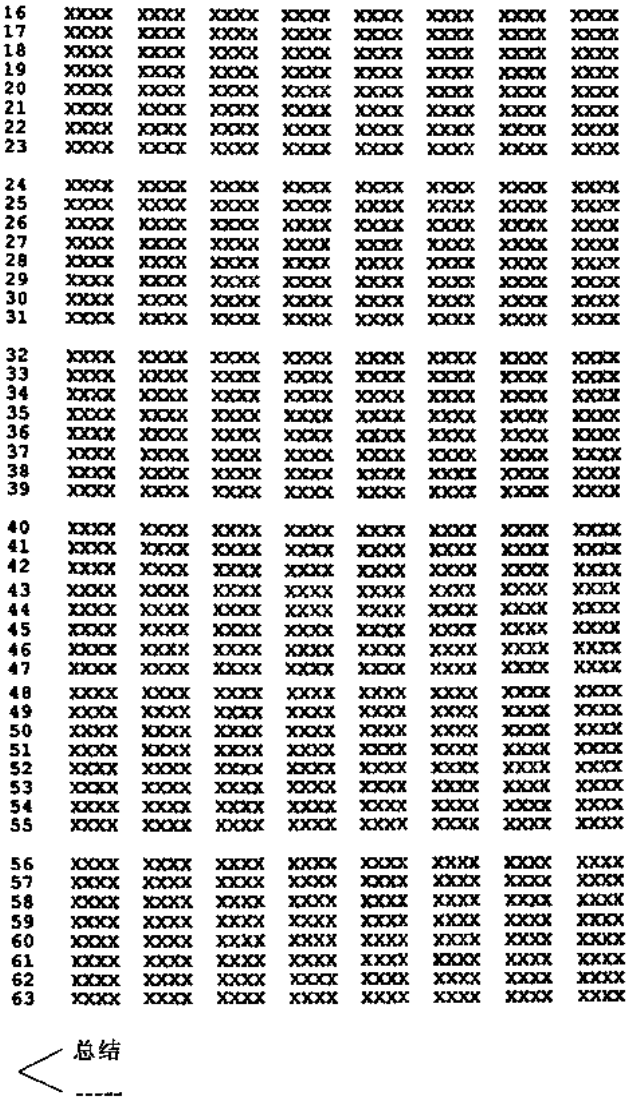


图 9-8

称作门阵(或全门阵列)的芯片包含非常大量的未连接门。它们往往包含的是相互连接产生专门的与非门或者非门的各个门阵。门阵的程序设计使这些门连接起来产生组合逻辑,触发器,寄存器,计数器,存储器等等。原始芯片做的容量非常大(因为此规模较经济),后来通过指定门之间的内部连接而成为定制式的。这些器件的程序设计要求相当复杂的软件开销和高度的计算机能力完成电路的布线和连接路径。有些设计规则,例如在计算机产生内部连接时限制交叉的数量,遵守这些规则常常使设计过程有很多次迭代。另外,内部互连一旦完成就不能改变。结果是门阵的使用被限制于高容量的产品并要求相对应的较长的开发周期。

有一种门阵列是制造的而不是用金属互连的。按定制设计编程,导致光刻掩膜布线图在制造过程由最后柜互连接这一步产生。这样的器件用几周时间就可完成送到最终的用户手中。

另一种类型门阵,是设计成栅格状的两个层,层之间互连,用户的程序转换成顶层金属模式即金属被腐蚀掉形成两层之间的相互连接。这种类型要求较少的制造步骤,减少了设计周期。

门阵列通常只使用少数几种标准晶体管单元电路,因此,就功率损耗或所有应用情况来说,其工作速度不可能做到完全优化。门阵列一般不像全定制那样有效地利用硅片资源。然

而,它可以制造得不太昂贵并比全定制设计更快。

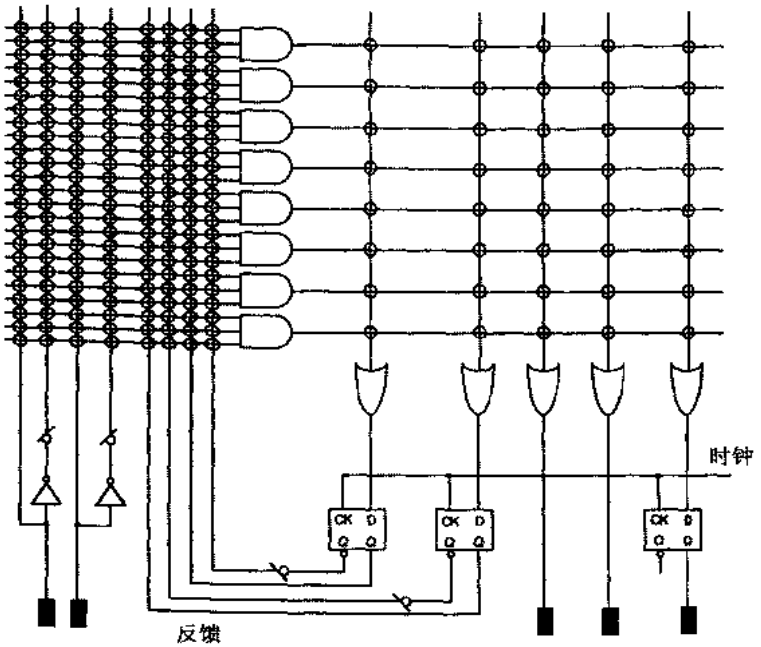


图 9-9 通用 PLA 结构

门阵列器件在计算机或工作站上与 PAL 或 PAL 器件相同的方式编程,但它要求相当大的软件和计算机功能。除处理连接和搜索之外,要考虑诸如过热点的发热问题。设计者的程序能以布尔方程形式输入,如前面所讨论的那样,或者用状态表方式输入。另一种通用编程设计方法即图形输入法。设计者通过在软件库中调用元件在工作站监视器产生逻辑图,并用鼠标或指针把它们连成所期望的图形。在形成连接程序和局部运算过程中,在宏中处理有固定名称码组合门(如寄存器)。用户往往会在软件中产生他们自己的宏。

门阵列编程产生的一套软件-生产计算机装置。因为照相平板印刷的艺术处理的实现有赖于在电路中形成所有门之间所需要的连接。门阵列经常被应用在超大规模集成电路芯片中实现完整的系统。

在一个设计被真正完成之前,其实现方式的优化是十分重要的。因为实验用的结构或电路板做起来会太贵;并且由于其尺寸的不一致而导致其不能精确预测时间的延迟;而且实现优化的主要工具只是第 6 章描述那种型式的模拟软件。几乎生成大型集成电路所有的设计软件都包括一个完善的模拟包,它能用于实现优化并在生产期间为应用提供测试信号。

9.7 可编程门阵列

一方面在 PAL 集成电路和 PLA 之间有些中间产品,另一方面全门阵列产品也被开发出来,叫可编程门阵列(PGA)或称为现场可编程门阵列(FPGA),和 PLA 相比它有相当少的内部固定结构,但比全门阵列的结构大得多。FPGA 有时叫逻辑单元阵列(LCA),是一种通过编程而不改变结构的 PLD。相反,用电控信号来改变固定电路的功能。FPGA 和控制电路在第 11 章讨论。

对 PGAs 的讨论是复杂的,因为实事上某些销售市场所谓 PGAs 产品应用连接技术永久地改变了器件,而不是像 LCA 那样通过编程来改变。把普通 PGA 器件看成 PLA 器件和全门阵列之间的中间产品也许是最恰当的。使用者应研究具体使用的产品结构和编程方法。

9.8 全定制设计

常有这种情况,在大容量高速度的应用中,客户的定制设计要求是制成一个完整芯片。这

是一个相当复杂的过程，涉及比较长的设计周期和熟练的设计训练。这种技术被广泛深入地研究并且创建了一个电子元件模型库。这些模型包括描述功耗的参数，速度性质，硅片利用和在制造过程中光刻掩膜技术的使用。模型可能是由各种分立的晶体管、电阻等，或它们能由门-级集成组成，也可能包含各种辅助系统宏，如在第 9.6 节提到的那样。如果电路设计使用高级宏和限定门组，就叫做标准单元设计。

习 题 解 答

9.1 图 9-10 所示是一个无板载触发器的通用 PLA，写出输出的布尔逻辑方程式。

解 ☞ 输入是 A, B, C, 而输出是 W, X 如图示。每个或门被一组与门驱动输入连接如图 9-10 所示的左侧，每个与门的组合输入都可用布尔表达式表示，例如：A′B′C′对应于最上面的与门。所有与项的组合连到每个或门：

$$W = A'B'C' + AB'C + ABC$$

$$X = A'BC + ABC' + ABC$$

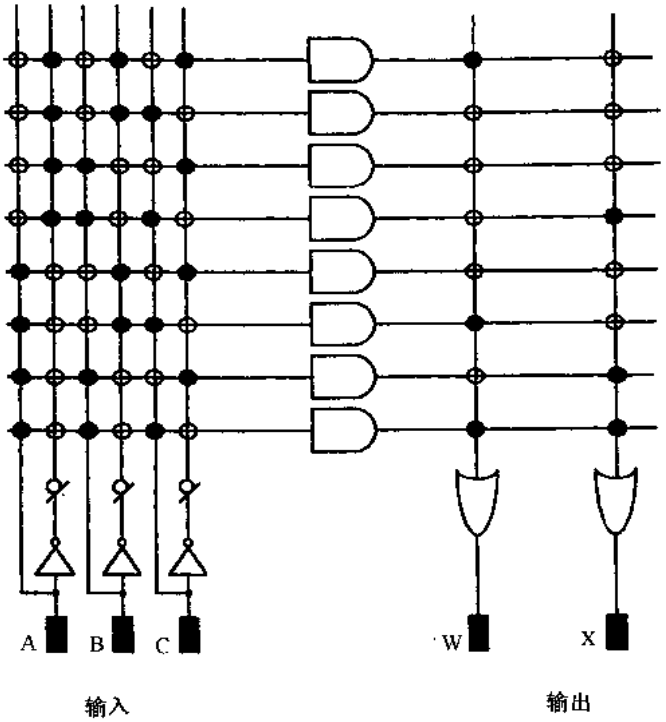


图 9-10

9.2 设计一个可反向的移位寄存器，使移位寄存器的每一位连接在环上，并符合下面的具体要求：

- 硬件是 16R6PAL，如图 9-4 所示。
- 当控制信号 LR 是 1 时，每个时钟脉冲使寄存器左移；当 LR 是 0 时，寄存器向右移。
- 在信号 LS 控制下，当其有效时，4 个输入端出现的数据并行装入移位寄存器；当其无效时，寄存器移位。

对布尔输入写出 PALASM 方程，并画出连接模式图。移位顺序如图 9-11 所示，指定系统框图如图 9-12 所示。

解 ☞ 1. 当 LS = 0 时，每个 FF 接收 d 输入，当 LR = 1 时，输入来自右侧邻近，当 LR = 0 时，来自左侧邻近。对触发器 K 使用布尔形式表示如下：

$$d_{in(k)} = (FF\ RIGHT)(LR) + (FF\ LEFT)(LR)'$$

2.在下载模式(LS=1)期间,寄存器的d输入被连到输入数据;当LS=0,连接应像上面第一步。写出布尔表达式:

$$d_{in(k)} = [(FF\ 右)(LR) + (FF\ 左)(LR)']'(LS)' + (K_{dat})(LS)$$

3.PALASM方程是:

$$A_: = D^*LR^*/LS + B^*/LR^*/LS + A^*_{dat}LS$$

$$B_: = A^*LR^*/LS + C^*/LR^*/LS + B^*_{dat}LS$$

$$C_: = B^*LR^*/LS + D^*/LR^*/LS + C^*_{dat} + LS$$

$$D_: = C^*LR^*/LS + A^*/LR^*/LS + D^*_{dat}LS$$

参考16R6图,在图9-13,显示了很多可能互连方案的一种。

这是考虑典型的时序逻辑电路,下面的两个问题被叫做状态机,在第10章中会更详细地讨论。

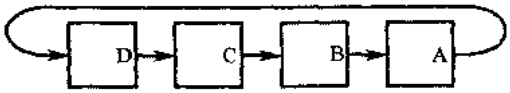


图9-11

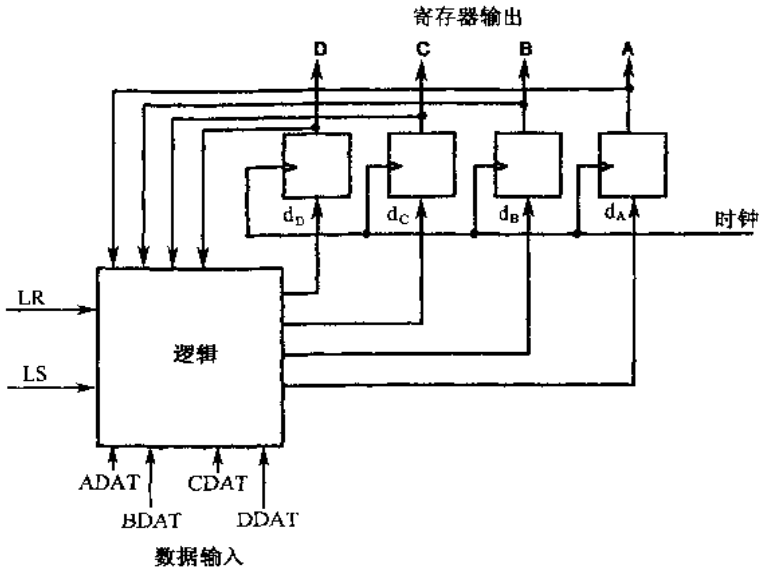


图9-12

9.3 一个简单的步进电机要求两个驱动电压(A和B),必须以指定的方式按顺序工作。每次在时钟脉冲发生前电压保持恒定,信号电压A和B如表9.2所示方式改变;并且每次电机轴向前转动1步。一个使电机转动的PAL6R6(布尔输入)程序在外部信号F/R命令下,正向或反向转动。PAL IC应该有3个输入:F/R,MOVE和时钟。

只有当MOVE是有效时,步进电机才运动,而旋转速率将由时钟频率决定。

表9.2 步进电机驱动电压

	A	B	
↓	0	0	↑
	0	1	
	1	0	
正向	1	1	反向

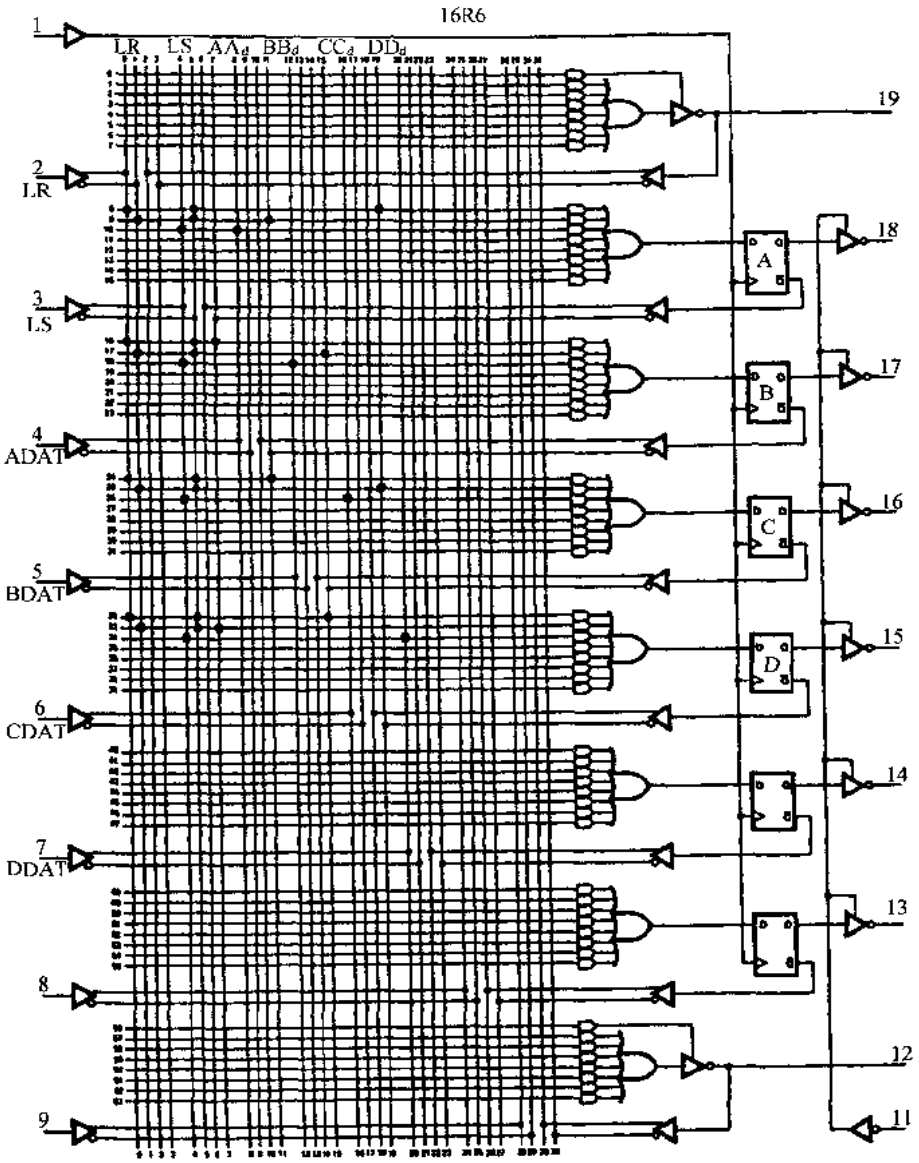


图 9-13　题 9.2 PAL16R6 电路连接模式

正向时序，在状态 10 后跟状态 00，逻辑 0 表示 0V，逻辑 1 表示适当的 dc 电压电平。

解 ☞ 下面将选择一种设计以合并 D 触发器和逻辑，其布尔方程描述将形成 PAL 器件输入编程。假设 F/R = 1，意味着正向移动，并且触发器 A 和 B 将被用于存储时钟脉冲之间的步进电机信号。

一个系统方框图如图 9-14 所示，每对 d 输入构成触发器次态，并且是在上层等待 1 个时钟脉冲点到其加到电机上并产生 1 个轴步。因此，在任何给定的时间，下一电机驱动电压 A 和 B 的应用取决于运动和方向信号，同时也取决于即在当前驱动信号的状态。逻辑必须被设计成能产生 D-触发器输入。该输入具有移动命令、方向信号 F/R 和当前触发器状态的功能。真值表(表 9.3)可以直接得到并符合所给出的要求。

注意，步进电机的输出 A,B 如何被用作逻辑输入去决定下一个状态与表 9.2 的时序信息相对应。当 M = 0 时，下一个状态输出不变，表示电机轴保持静止。

从真值表可以直接写出 PALASAM 形式的布尔方程：

$$A: -/F^*/M^*A^*/B + /F^*/M^*A^*B + /F^*M^*/A^*/B + /F^*M^*A^*/B$$

$$+ F^*/M^*A^*/B + F^*/M^*A^*B + F^*M^*/A^*B + F^*M^*A^*B$$

$$B_i := /F^*/M^*/A^*B + /F^*/M^*A^*B + /F^*M^*A^*/B + /F^*M^*A^*B + F^*/M^*/A^*B$$
$$+ F^*/M^*A^*B + F^*M^*/A^*/B + F^*M^*/A^*B$$

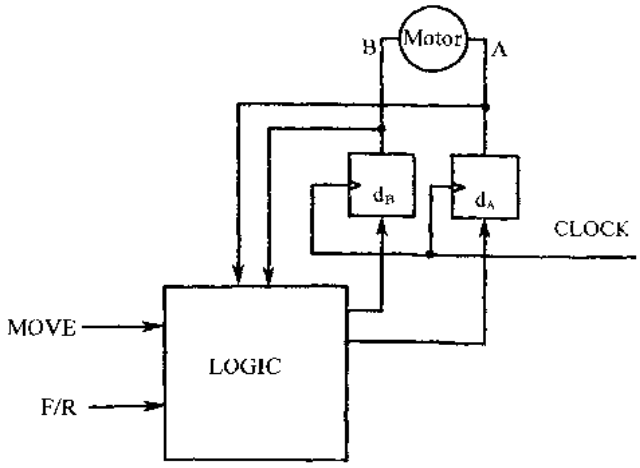


图 9-14

表 9.3

逻辑输入				输出(次态)	
F/R	M	A	B	d_A	d_B
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	0

PALASAM 符号(:=)用来表示输出 A 和 B 被寄存。实际上,这些变量出现在他们相关方程两边,表示下一个状态值是当前值的函数,如上面描述的那样。

对每个或有 8 个与项,只要求两个 D 触发器。这样,这个设计就适合于 16R6,而不需简化。

9.4 希望实现表 9.4 所示的时序,用 PAL16R6 中的触发器。当控制输入 U/D = 0,顺序向上,而当 U/D = 1,方向相反,运动向下。问用给定的 PAL IC 是否适用。如果可以使用,写出控制逻辑方程,同时指出如何使触发器初始化为 0000。

表 9.4

D	C	B	A
0	0	0	1
0	1	0	1
1	0	1	0
1	1	0	0
1	1	1	0
0	0	1	1
1	1	0	1
0	0	0	0

向上　　向下

重复

解 像前面的问题一样，当接收时钟脉冲时，触发器输出才改变状态，假设用当前状态和 U/D 变量作为输入，由逻辑电路决定下一个状态，由给定的时序可以写出一个真值表。例如，从表 9.4 看出，当 U/D=0 时，当前状态为 1110 时(第 5 行)，则下一状态将为 0011，而当 U/D=1 时为 1100，结果表示在表 9.5，没有发生的状态已被忽略掉。

表 9.5

逻辑输入（现态）					逻辑输出（次态）			
UD	D	C	B	A	d_D	d_C	d_B	d_A
0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	1
0	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0
0	1	1	0	0	1	1	1	0
0	1	1	0	1	0	0	0	0
0	1	1	1	0	0	0	1	1
1	0	0	0	0	1	1	0	1
1	0	0	0	1	0	0	0	0
1	0	0	1	1	1	1	1	0
1	0	1	0	1	0	0	0	1
1	1	0	1	0	0	1	0	1
1	1	1	0	0	1	0	1	0
1	1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	0	0

因为每个输出变量不多于 8 个 1，所以，对应 8 个输出有 8 个与门就够了，在 16R6 上，得到的结果是数字。有 4 个输出变量选择 4 个触发器，因为这些都可得到，所以设计能满足要求。

直接从真值表得到方程：

$$D_i = (U/D)'D'C'BA + (U/D'D'CB'A + (U/D)'DC'BA' + (U/D)'DCB'A' + (U/D)D'C'B'A' + (U/D)D'C'BA + (U/D)DCB'A' + (U/D)DCBA'$$

$$C_i = (U/D)'D'C'B'A + (U/D)'D'C'BA + (U/D)'DC'BA' + (U/D)'DCB'A' + (U/D)D'C'B'A' + (U/D)D'C'BA + (U/D)DC'BA' + (U/D)DCBA'$$

$$B_i = (U/D)'D'CB'A + (U/D)'DCB'A' + (U/D)'DCBA' + (U/D)D'C'BA + (U/D)DCB'A' + (U/D)DCB'A$$

$$A_i = (U/D)'D'C'B'A' + (U/D)'D'C'B'A + (U/D)'D'C'BA + (U/D)\ DCBA'$$

$$+ (U/D)D'CB'A + (U/D)DC'BA' + (U/D)DCB'A$$

同样，要求初始化，用信号 IN 和所有积项与，当 IN 是 0 时，D 输入到触发器是 0，当下一个时钟脉冲到达时，所有触发器为 0 态，当 IN 为 1 时，由逻辑方程定义与门为空方式，用 PAL16R6 实现的连接图，如图 9-15 所示。

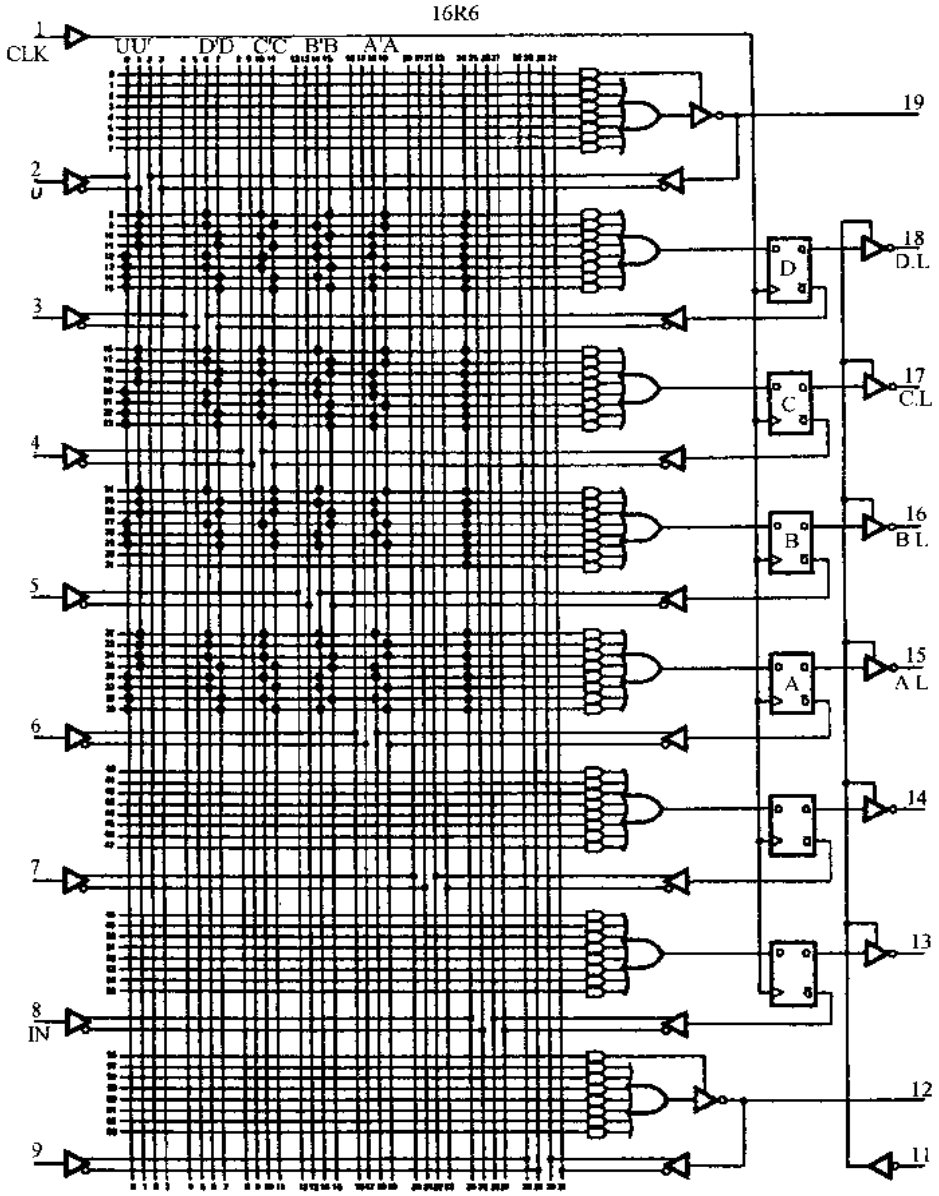


图 9-15 题 9.4PAL16R6 电路连接模式

9.5 对图 9-16 所示的通用 PAL 逻辑电路图，图 9-17 所示的信号被加于各输入端上。画出输出 A, B, C, D 和 E 端的输出波形，忽略内部传输延迟，并假设所有触发器都在第 1 个时钟脉冲前被清零。触发器为上升沿触发的 D 触发器(见第 7 章)

解 1.从图 9-16 看到触发器输入方程为

$$d_A = B'X + BY \qquad d_B = AX + A'Y \qquad d_E = AB' + A'B$$

2.C 和 D 内部不连接，并不影响触发器的状态。

3.初始化 A, B 和 E 为 0(清零)并且 X 和 Y 是 0(给出波形)

当时钟脉冲发生时，得到真值表表 9.6。

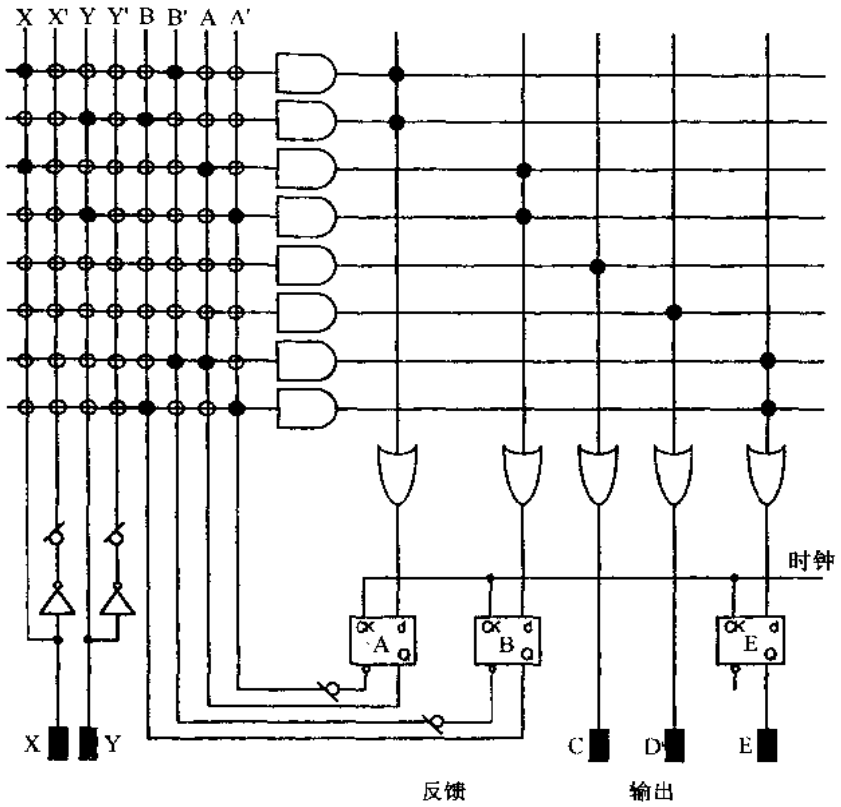


图 9-16

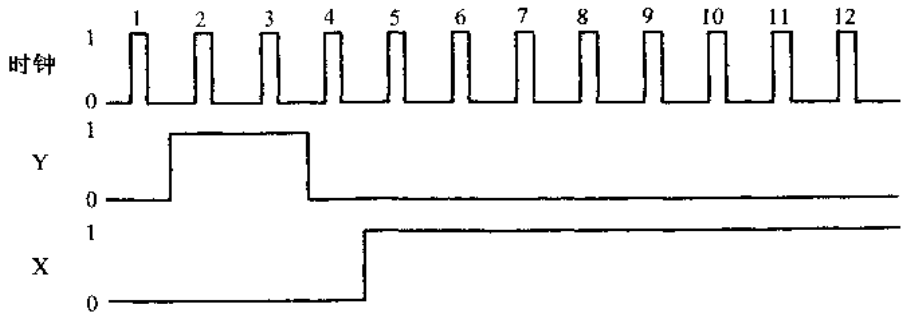


图 9-17

在表 9.6 中读下列 A, B 和 E 得到所要求的波形，结果表示在图 9-18。

表 9.6

			现态		次态			
脉冲	X	Y	A	B	d_A	d_B	d_E	E
1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0
3	0	1	0	1	1	1	1	0
4	0	0	1	1	0	0	0	1
5	1	0	0	0	1	0	0	0
6	1	0	1	0	1	1	1	0
7	1	0	1	1	0	1	0	1
8	1	0	0	1	0	0	1	0
9	1	0	0	0	1	0	0	1
10	1	0	1	0	1	1	1	0
11	1	0	1	1	0	1	0	1
12	1	0	0	1	0	0	1	0

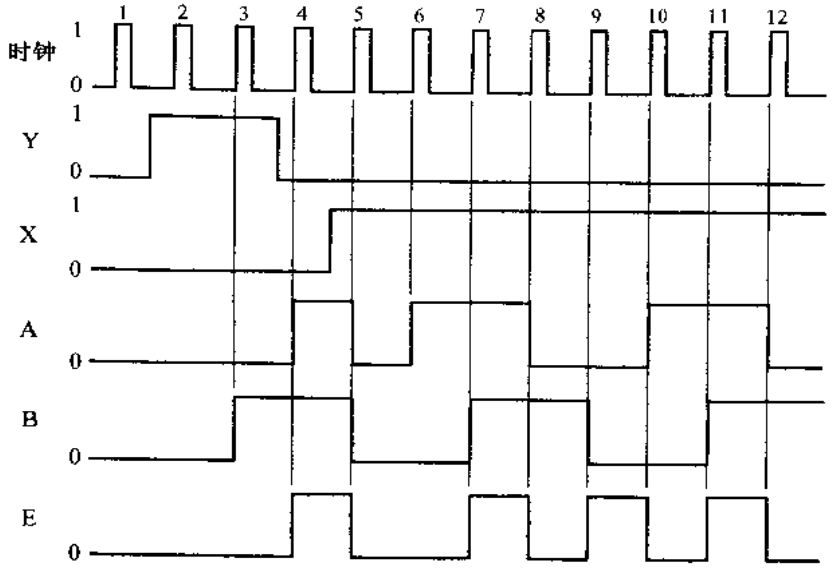


图 9-18

9.6 图 9-19 所示的 PAL 器件电路图，包含有两个触发器和组合逻辑的连接。假设信号 IN 初始化为无效(low)，并且，因为它在 t=0 时，跟着第 1 个时钟脉冲立即变成真。求触发器的时序并表示成一系列十进制数，假设 A 是最高位，跟在其后的是 B, C, D 位，求输出 E 何时变为真。

解 因为无与门连接到 E，输出 A, B, C 和 D 与它无关，因此可将 E 看成一个单独的位处理，其逻辑关系为

$$d_A = AB' + AC' + A'B \qquad d_C = B'C + A'D' + A'C' + ABC'$$

$$d_B = AC' + B'D + B'C \qquad d_D = A'B' + BD'$$

可变换成图 9-20 所示。

由于 IN 被初始化为无效，并且，由于它连接到所有与门，驱动触发器的第 1 个时钟脉冲读 0000 到 ABCD。这种情形用图的左上角表示。从图 9-20 对应于输入，集中表示 ABCD 的下一个状态，看成是 ABCD=0011。同样，跟着第 2 个时钟脉冲，图形变成第 1 行第 3 列，在这点输入表示 ABCD=0111 的次态将发生在时钟脉冲 3。这个过程能够继续，直到得到的值和前面遇到过的值重复为止，表示时序重复(参考表 9.7)。

连接到输出触发器 E 的是 A'B'C'D'(ABCD = 0000)和 A'CD(ABCD=0111 或 0011)。这样，E 在计数的前 3 个时钟周期才有效。

表 9.7 时序 0-3-7-8-12-15-0 的表示

A	B	C	D
0	0	0	0
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	0
1	1	1	1
0	0	0	0

重复

9.7 使用第 2.4 节中讨论过的 1 位二进制加法器，说明对于图 9-21 所示的通用 PAL，如何从真值表直接编程。令每个与门对应真值表中的 1 行而每个或门和输出的列相对应。

解 按表 9.8 所示逻辑，与门的编程按真值表的行顺序（如图 全译码器那样）然后连接每一项（行）到相应的输出或门，结果如图 9-22 所示。注意：真值表 1 和在图中所做连接的对应。

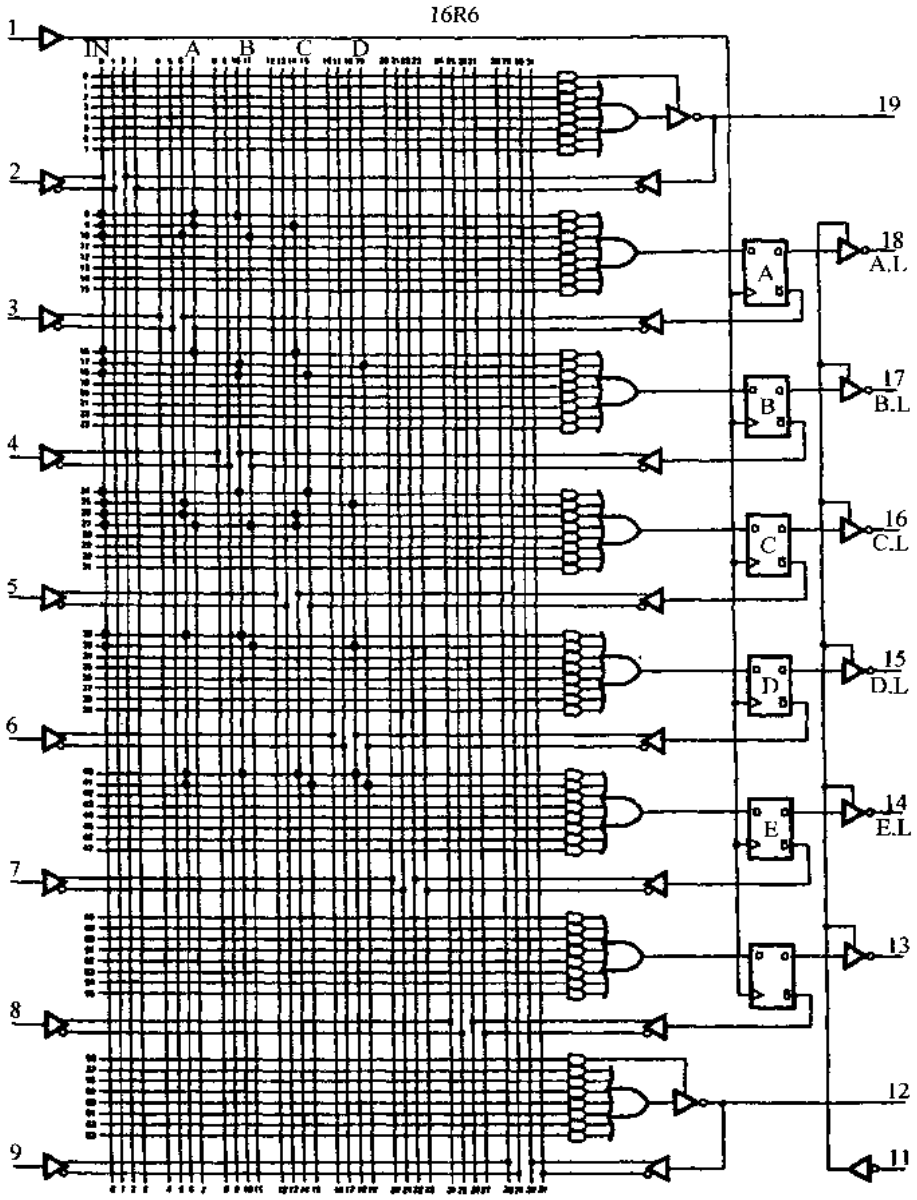


图 9-19　题 9.6 的 PAL16R6 电路连接图

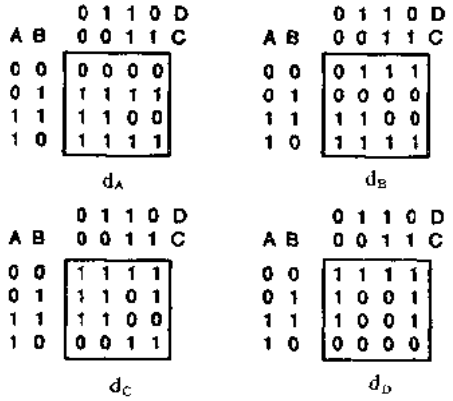


图 9-20

表 9.8

X	Y	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

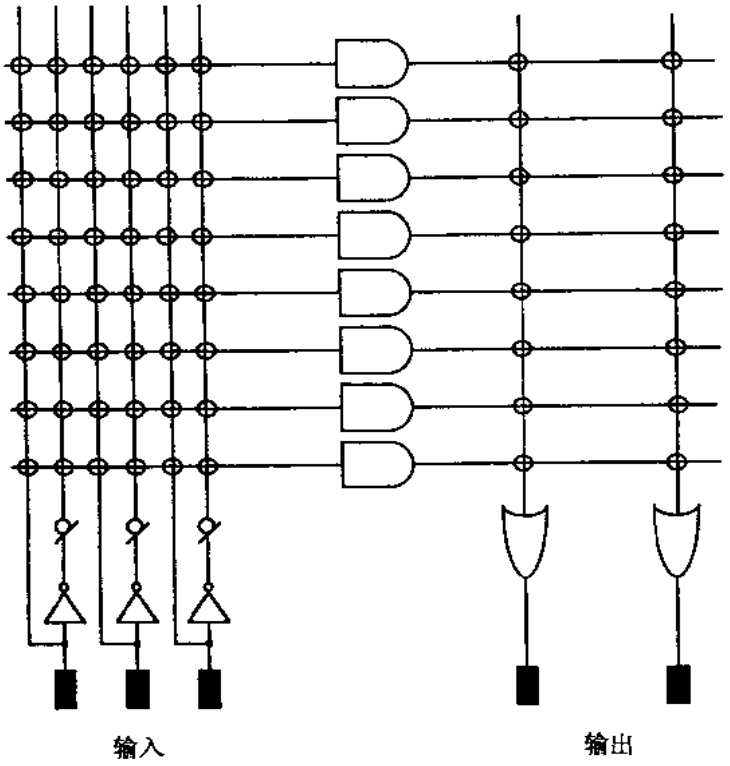


图 9-21

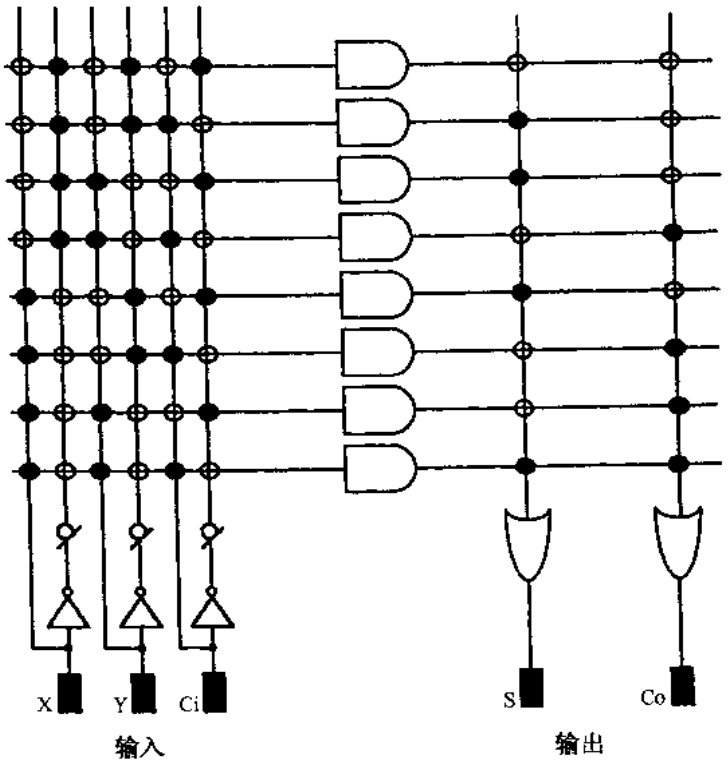


图 9-22

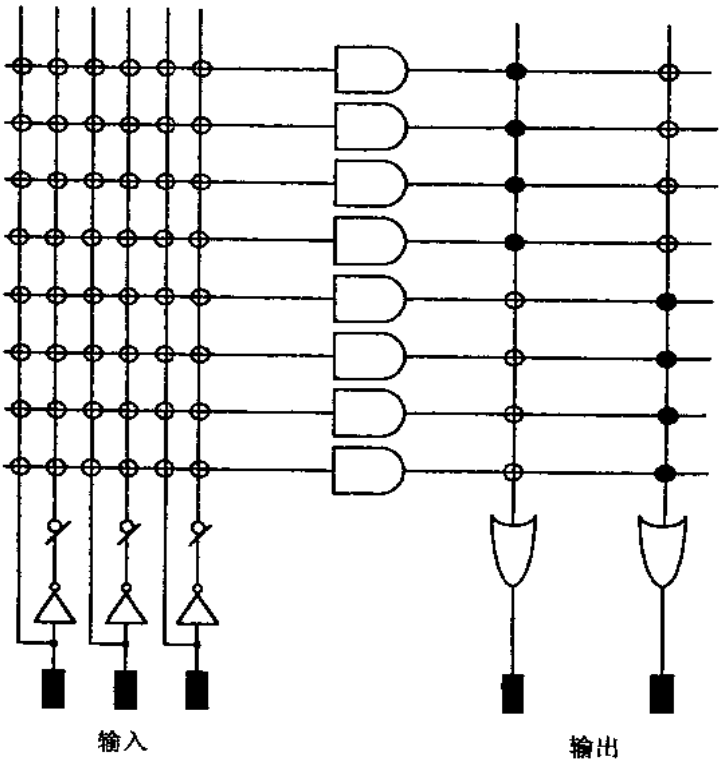


图 9-23

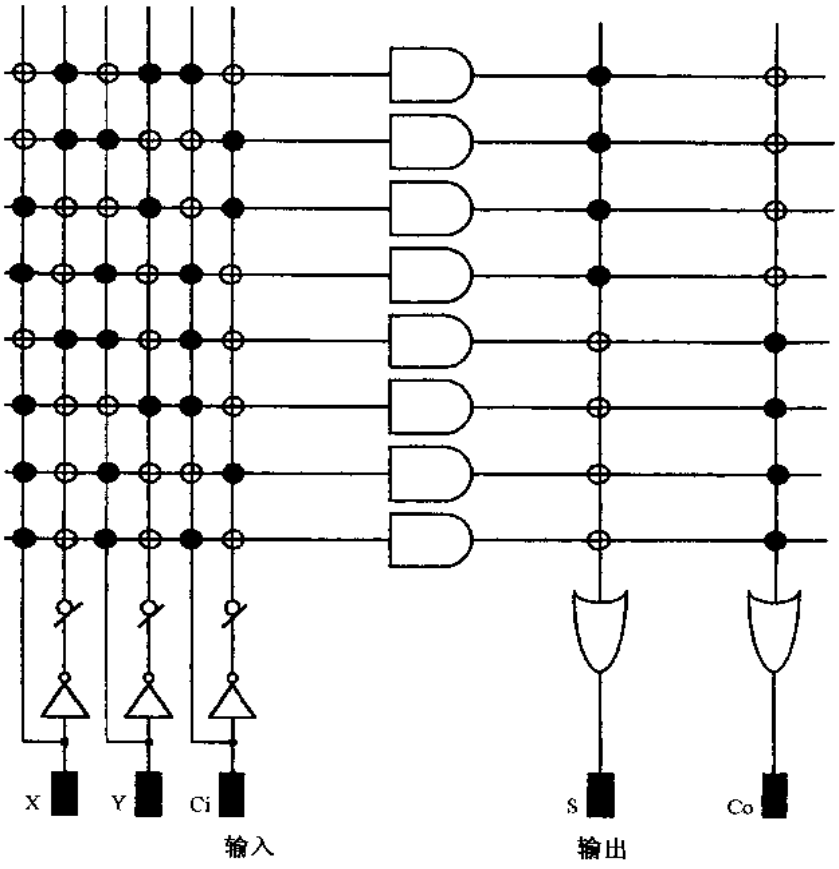


图 9-24

9.8 在通用 PAL 上实现问题 9.7 的加法器，如图 9-23 所示。说明它从一个标准真值表直接编程是不可能的，但有实现功能的可能性。

解 提示：PAL IC's 的结构，与门是成组的，并且每组都与 1 个或门的输入项连接，只有当全部与门的组连接到真值表的输入，才有对应关系。在该问题的解中，与门必须重新按真值表的关系排列。

真值表中十进制的 1，2，4 和 7 行组合到输出 S，而译码器输出对应于 3、5、6 和 7 行，被组合到输出 C。（第 2 个或门）由于连接表示 1，我们看到真值表中为 1 的行被重新排列，其中第 7 行用了两次（图 9-24）。

补充习题

9.9 图 9-25 给出了通用可编程 PAL 器件，写出输出为 C，D 和 E 的逻辑方程。

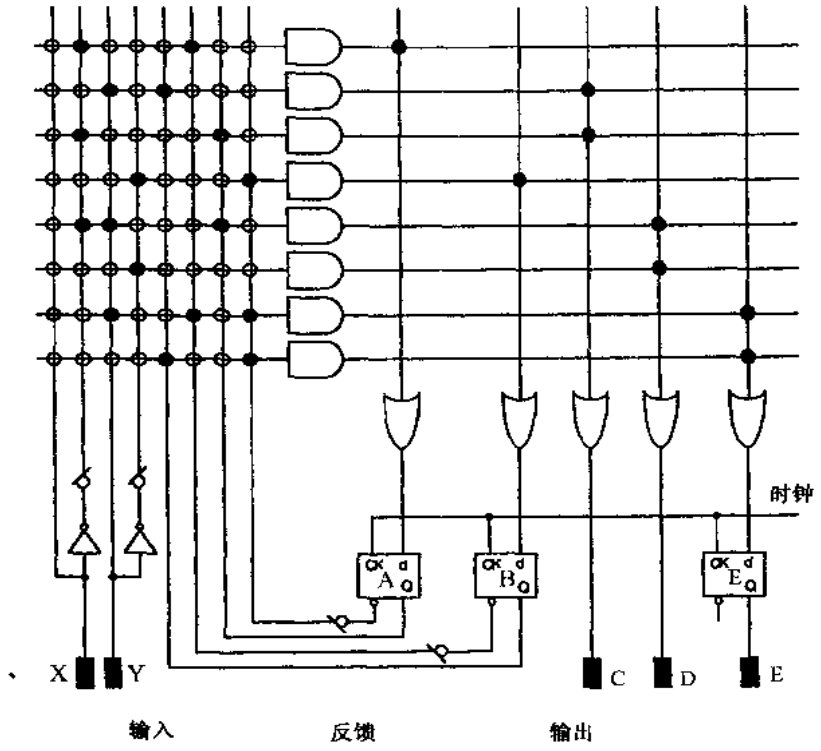


图 9-25

9.10 使用图 9-25 的系统，决定时钟工作的时序使给定的 X，Y 保持在 0，A 和 B 的初始值全为 0。

9.11 上题中对时序的影响是什么？如果在第 3 个时钟脉冲之后而在第 4 个时钟脉冲之前，Y 跳到 1，并且维持在那儿。

9.12 对图 9-26 所示是 PAL IC 连接成为一个计数电路，假设 11 端接到低位触发器初始化为 0，按时间顺序写出 ABCF 的等效输出值，因为很多个时钟脉冲要求重复发生。

9.13 对于 PLA 器件的连接模式，说明借助简单的 16 进制码，怎样来严密定义。参考图 9-27 所示的通用 PLA，每行可能的相互连接。用 3 位数 16 进制码描述，前两个数字表示 8 个输入行，后一个数表示 4 个输出行，用行码描述图 9-27 的连接。

9.14 参考题 9.13，是否有不允许的 16 进制码？

9.15 对于题 9.13 的 PLA，指出下面 16 进制码字的逻辑定义。

49A

4AA

86A

85A

454

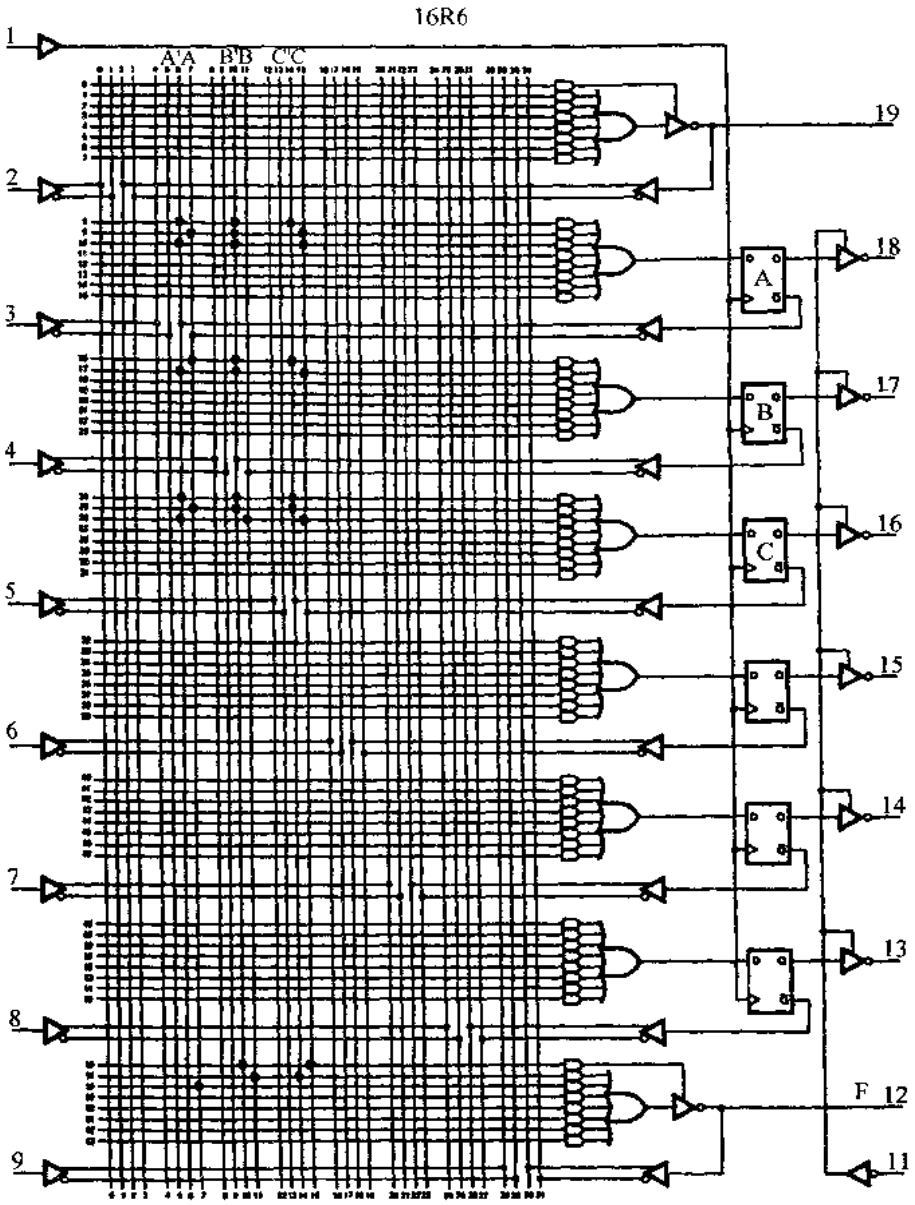


图 9-26　在 PAL16R6 电路图上画出题 9.12 的连接模式

494

864

8A4

9.16　使用题 9.15 的 6 个程序，列出 PLA 的时钟执行表，假设 A 和 B 是初始化状态为 0，

(a)X 总是 0。

(b) X 总是 1。

9.17　在 PAL 16R6 上画连接图，实现同步可逆计数器，其时序如表 9.9 所示。当控制输入 F 为 1 时，使计数器为减法计数；当 F=0 时，电路为加法计数。

9.18　使用 PAL 16R6 器件作为硬件，要求实现表 9.10 的时序，表示出所有的连接。假设寄存器初始化状态为 0011。

9.19　如果题 9.18 的计数器从 ABCD=0001 开始，下一个时钟脉冲到来之后，寄存器输出将如何？

9.20　表 9.11 所示的计数时序是在 PAL 16R6 器件上实现的。写出 PALASM 方程所要求的程序设计。

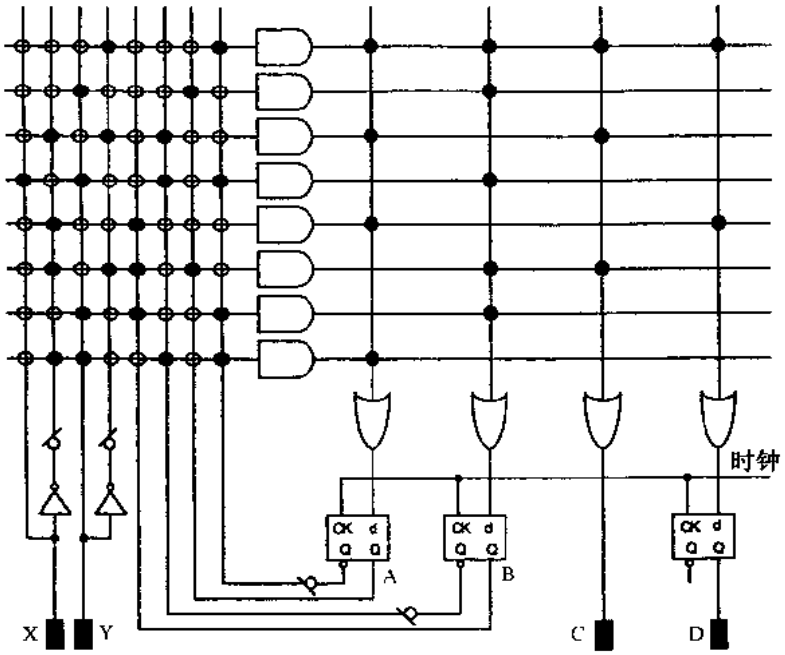


图 9-27

表 9.9

A	B	C	D
0	0	0	0
0	0	1	1
0	1	1	1
1	0	1	0
1	1	1	0
0	0	0	1
0	1	0	1
1	0	0	0
1	1	0	0
1	1	1	1
0	0	0	0

UP ↓ DOWN ↑

表 9.10

A	B	C	D
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

表 9.11

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0
0	1	0	1
1	0	1	0
1	0	1	1
1	1	0	1
1	1	1	0
1	1	1	1

9.21 参考题 9.20 分别使用寄存器元件 D, C, B 和 A 的输出端 18, 17, 16 和 15, 假设在或门组内, 与门按数字顺序安排并表明在 PAL 16R6 图上可能的连接。

补充习题答案

9.9 $C = YB + X'A$; $D = X'YA + Y'$; $d_E = A'B + A'B'Y$

9.10 见表 9.12

第 1 个时钟脉冲之后, 当 D 和 E 保持在 1 和 0 时, C 将同步地在 1 和 0 之间(振荡)变化。

9.11 见表 9.13, 第 6 个时钟脉冲之后, 不发生变化。

表 9.12

时钟#	C	D	E	
0	0	1	–	←不确定
1	1	1	0	
2	0	1	0	
3	1	1	0	

表 9.13

时钟#	C	D	E
4	0	0	0
5	1	1	1
6	1	1	0
7	1	1	0

9.12 0, 11, 9, 6, 3, 13, 0.

9.13 11F

224

54A

A54

489

5A6

294

658

提示: 有关连接的实际信息如何被转换为一组 16 进制数, 用数字计算机可以比较容易地处理。

9.14 由于数字信号和它的逻辑反不应该同时接到同一与门的输入上, 6 个字的前两个数字不可能包含 3, 7, B, C, D, E 或 F, 第 3 个 16 进制数没有这样的限制。

9.15 $d_A = C = X'A'B - X'AB + XAB' + XA'B'$

$d_B = X'A'B' + X'A'B' + XAB' + XAB$

$d_D = 0$

9.16

(a)

A	B
0	0
0	1
1	1
1	0

(b)

A	B
0	0
1	0
1	1
0	1

9.17 用指定的 PAL 器件, 只可能有 8 个与项, 而真值表要求 10 个; 这样, 将设计转换成为熔丝图之前, 要求使用某一种简化方法才能满足。

PAL 16R6 连接图表示在图 9.28。答案是

$$A_N = F'AB' + F'AC' + F'A'B + FAB + FA'B'C'$$

$$B_N = F'AB' + F'AC' + F'B'D + FB'C' + FAD + FB'D'$$

$$C_N = F'A'D' + F'B'C + F'A'C + F'ABC' + FCD' + FA'BC + FA'B'C'$$

$$D_N = F'A'B' + F'BD' + FB'D' + FA'B$$

9.18 如用 K 图化简

$$A_N = AB' + BCD$$

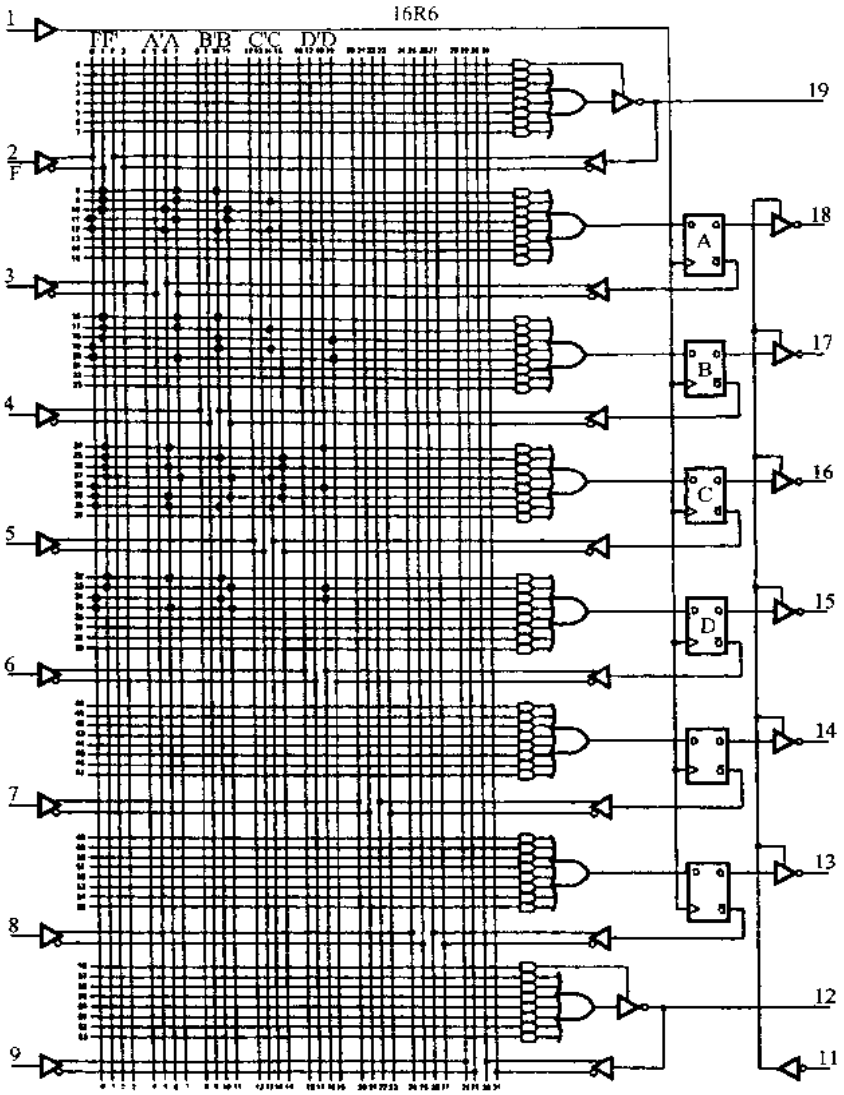


图 9-28 在 PAL16R6 电路图上题 9.17 的连接模式

$$B_N = A'D' + A'C' + B'CD$$

$$C_N = AB + C'D + CD'$$

$$D_N = D'$$

连接图表示在图 9-29。

9.19 如所有不确定值被认为是0(非"无关最小项"),下一个状态将是0000。如果"无关最小项"在K图中用1s包含,如题9.18给出的解答,下一个状态是0110。

9.20 $D_1 = /D^*C^*/B^*A + D^*/C^*B^*/A + D^*/C^*B^*A + D^*C^*/B^*A + D^*C^*B^*/A$

$C_1 = /D^*/C^*B^*/A + /D^*C^*/B^*/A + D^*/C^*B^*A + D^*C^*/B^*A + D^*C^*B^*/A$

$B_1 = /D^*/C^*/B^*A + /D^*C^*/B^*A + D^*/C^*B^*/A + D^*C^*/B^*A + D^*C^*B^*/A$

$A_1 = /D^*/C^*/B^*/A + /D^*C^*/B^*/A + D^*/C^*B^*/A + D^*/C^*B^*A + D^*C^*B^*/A$

9.21 见图 9-30。

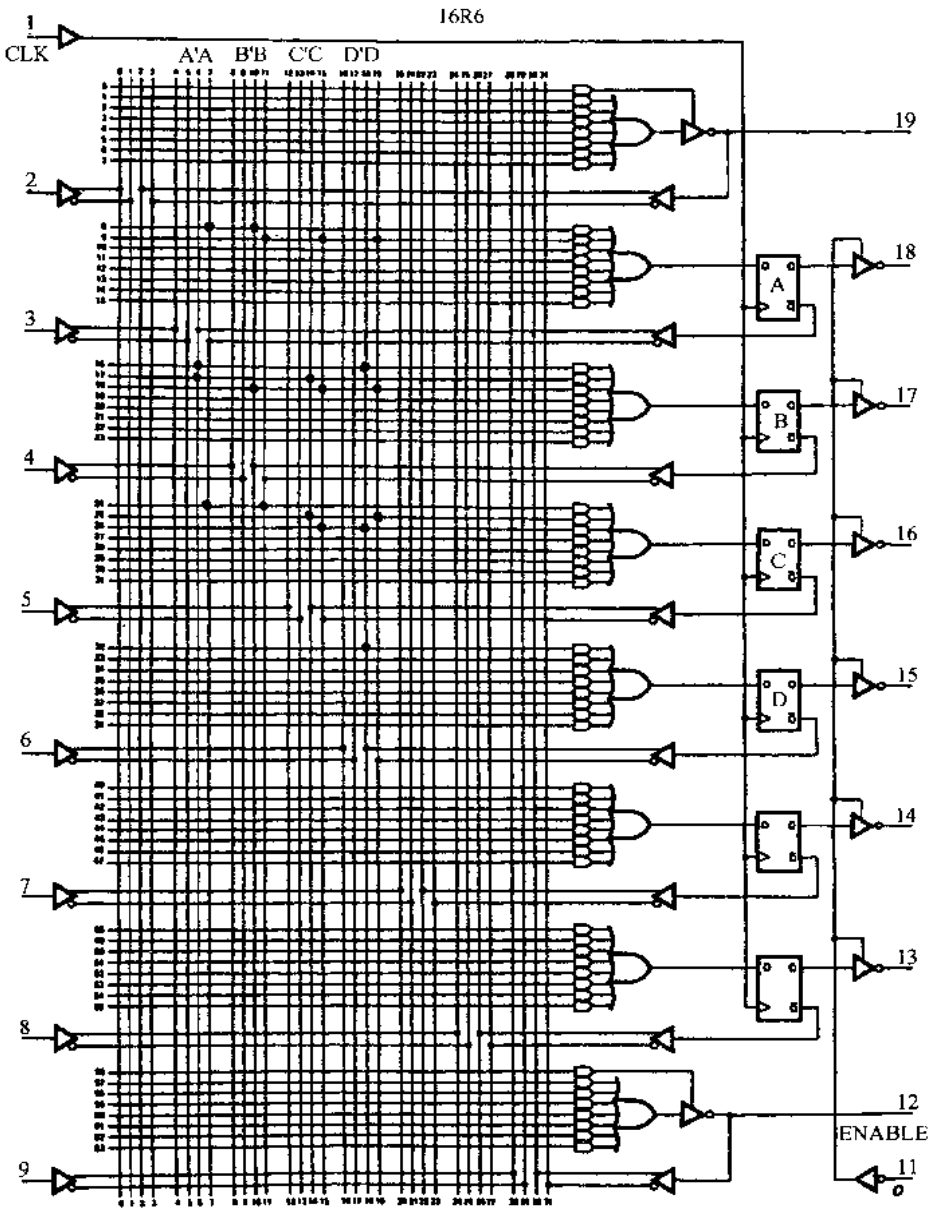


图 9-29　在 PAL16R6 电路图上题 9.18 的连接模式

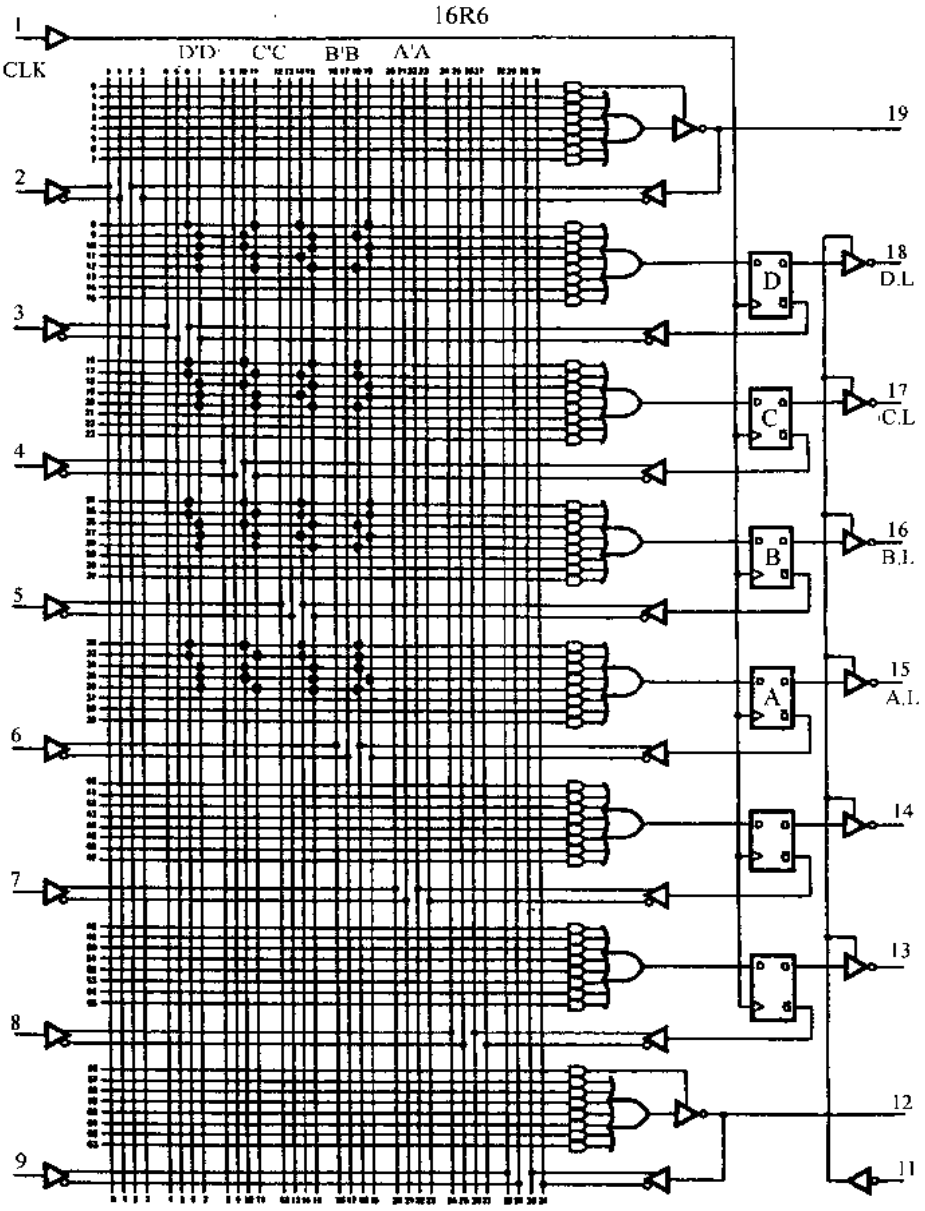


图 9-30 在 PAL16R6 电路图上题 9.21 的连接模式

第 10 章　简单状态机设计

10.1　引言

状态机是构成时序数字逻辑电路中重要的一种，具有以下的性质：

1. 它包含的组合逻辑部分有明确的意义。
2. 有同步时钟锁存的寄存器，其逻辑输出能直接使用或能反馈到组合逻辑输入。
3. 组合逻辑有接收能力，如输入或者反馈信号或外部输入（或二者兼有）。它产生的输出可经外端子输出，和/或用于锁存寄存器的输入。

状态机的基本组成部分如图 10-1 所示。

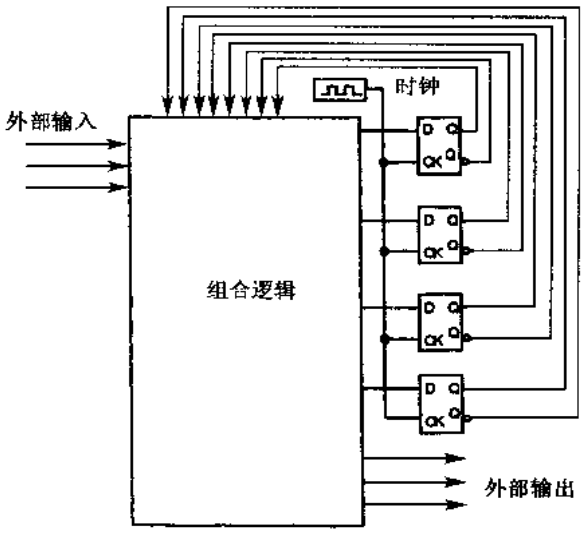


图 10-1　简单状态机

组合逻辑的目的是在接收每个时钟脉冲之前发出命令，这些命令要求状态机执行其预备功能。

状态机逻辑和寄存器部分的精确划分遵循其设计的简化组合逻辑问题。状态机的一个重要特征是同步时钟定时输入到寄存器，确保组合逻辑电路的时域差别不影响其特性。

10.2　用 D 触发器设计的传统状态机

状态机设计的传统方法，是使用基本系统特性的真值表。当前的或系统的现状态存储在寄存器触发器中。组合逻辑接受现状态数据作为输入（可能和外部输入信号一起）并产生表示寄存器的 D 输入的次态。在新的接收时钟脉冲之后，数据就被锁入寄存器变为新的现态，并且重复此操作过程。因为下一个状态的数据在新的时钟脉冲到来之前是静止的，因此时序状态机设计能简化成组合逻辑的设计问题，在此，真值表可用于指定触发器将来的状态以作为现态变量值的函数。

例 10.1　同步计数器的特性。

在同步计数器设计中，计数和状态是统一的。考虑到 4 位格雷码计数器，其计数顺序如表 10.1 所示。寄存器的各触发器用 D, C, B 和 A 表示。

这个时序很容易转换成现态/次态真值表:(表 10.2(a)和(b))。

状态表(表 10.2(a))可直接从计数顺序得到,并能转换成通常的逻辑设计真值表(表 10.2(b))。注意:状态到状态跃变的工具是组合逻辑,由现态/次态真值表指定的,而且在第 3 章到第 5 章中描述的设计方法的任何一种方法均可用来得到它,并和第 9 章讨论的使用可编程器件逻辑设计方法一起。几种方法都将在本章的习题中继续讨论。

如果需要外部输入,附加变量必须加在真值表的输入侧,实际上使真值表长度增加一倍。如果需要一个不是从寄存器的触发器来的输出,那么,附加输出变量必须加到真值表中。

表 10.1　计数顺序

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

重复

例 10.2　假设一个外部正 / 反向命令(F/ R)被引入例 10.1 的系统中。变量 F/ R 的功能是改变状态到状态的跳变顺序。如 F / R 是 0,电路计数顺序表示在表 10.1 中;如 F / R 是 1,按相反顺序计数(即从底到上)。另外,假设辅助输出被指定对次态提供奇校验位。写出新的逻辑设计真值表。

表 10.2(a)　从计数顺序得到状态表

状态							
现态				次态			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	0
0	0	1	0	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	0	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	0	0	1	1	0	0	0
1	0	0	0	0	0	0	0

表 10.2(b)　对应的真值表

(按二进制顺序输入)

输入(现态)				输出(次态)			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

表 10.3(a)　逆计数顺序状态表

状态							
现态				次态			
D	C	B	A	D	C	B	A
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1
0	0	1	0	0	0	1	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	1	0
0	1	0	1	0	1	1	1
0	1	0	0	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	1	1	0	0
1	1	1	1	1	1	0	1
1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0
1	0	1	1	1	0	1	0
1	0	0	1	1	0	1	1
1	0	0	0	1	0	0	1

表 10.3(b)　逆计数真值表

输入(现态)				输出(次态)			
D	C	B	A	D	C	B	A
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	1
1	0	1	0	1	1	1	0
1	0	1	1	1	0	1	0
1	1	0	0	0	1	0	0
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	0	1

对于反向计数情况，其顺序必须转换成状态表，然后重新排列成逻辑设计的真值表(象例10.1那样)。表 10.2(b)和表 10.3(b)组合成一组，在最高位列，变量 F/ R 被加到输入侧，奇偶输出列如表 10.4 所示。

表 10.4　带奇偶校验的可逆计数器真值表

输入(现态)					输出(次态)					输入(现态)					输出(次态)				
F/R	D	C	B	A	D	C	B	A	P	F/R	D	C	B	A	D	C	B	A	P
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	1
0	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	0	1	0	0	1	1	0	0	1	1	0	1	0	0	0	1	0	1	1
0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	1	1	0
0	0	1	1	0	0	1	1	1	0	1	0	1	1	0	0	0	1	0	0
0	0	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	0	1
0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	1	1
0	1	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0	1	1	0
0	1	0	1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	0	0
0	1	0	1	1	1	0	0	1	1	1	1	0	1	1	1	0	1	0	1
0	1	1	0	0	1	1	0	1	1	1	1	1	0	0	0	1	0	0	0
0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	1
0	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	0

现态/次态真值表形成组合逻辑及其相关的同步状态机设计的基础，这种逻辑由前几章讨论过的任何方法都能实现。如果用小规模集成电路门能实现，那么对每个输出变量可得到一个 5 变量的卡诺图，化简后，用有效的硬件就能实现。更好的方法是使用多路选择器，对每一个输出变量编程。

另一种选择就是使用可编程逻辑器件，如 PALIC 或 PLA。这种情况，可通过使每一个输出变量进入到布尔方程，从而实现此逻辑。小的 PAL 装置是二者中较小的，应该首先用来试试这儿，主要设计是决定可编程器件的适当规模。如果奇偶校验输出被寄存，则在线路板上最低 5 位的触发器一定要有。对于从逻辑方程得到的单个或最大的与项数，也许是使用卡诺图得到的，要和包含在芯片中的最大与项数比较。另一方面，方程应该从真值表直接输入，没有经过化简，并且软件上将化简逻辑，并通知用户设计是否使芯片过大。在这种情况下，考虑用较大的 PALIC 或者用 PLA 实现。

10.3　用 J-K 触发器设计

在状态机设计中，设计者常常在 D 和 J-K 触发器之间进行优化选择，这种选择取决于具体的应用。下面的例子说明了设计方法上的不同。

例 10.3　考虑表 10.5 中所给的计数器。状态表已经写出(表 10.6)，对每一个输出变量的次态图都可从此状态表得到。利用“无关最小项”状态推导出适当的布尔方程，如图 10-2 所示。图中坐标表示每种可能的现态，输入到给定的位置，表示出对应的变量次态值。例如，如果现态是 DCBA＝1100，从 D 图看到下一个时钟脉冲 D 就变为 1；从 A 图看到，次态值 A 为 0。因为未能在给定序列把所有可能的情况包含进来，显然用无关最小项表示这些状态(用“＊”表示无关最小项)。

表 10.5

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	1	0	0
1	1	1	0
1	1	1	1

重复

(表示向下顺序进行)

表 10.6

现态				次态			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

D 触发器实现

图 10-2 的布尔方程可用来实现组合逻辑，在图 10-1 中，其输出驱动 D 触发器的输入。

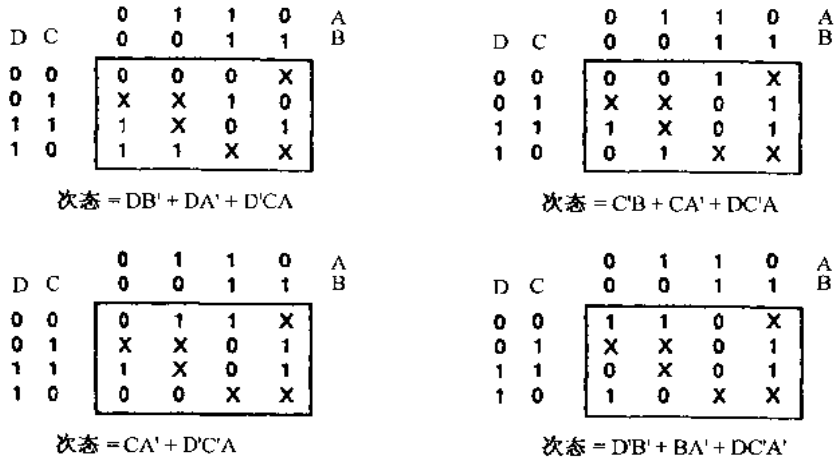


图 10-2

JK 触发器实现

考虑第 7 章(表 10.7(a))JK 同步真值表。如果输出 Q 是 0，在时钟之后它就保持在 0，如果 JK = 00 或 JK = 01，即它只要求 J = 0，K 的值是无关的。如果输出 Q 是 0，且 JK = 10 或 JK = 11(K 的值还是无关的。)在时钟脉冲之后它就变成 1，如果 Q = 1，且 JK = 01 或 JK = 11(K 必须是 1，而 J 是无关的)，它将变成 0；如果 Q = 1，且 JK = 10 或 JK = 00(K 必须是 0，忽略 J)，它就保持为 1。

如表 10.7(b)所示，这个特性可能综合在时序-状态-变化表中。

表 10.7(a)　JK 真值表

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q'_n

表 10.7(b)　JK 时序-状态-变化表

变化	J	K
0 to 0	0	X
0 to 1	1	X
1 to 0	X	1
1 to 1	X	0

JK 时序-状态-变化表和次态图 10-2 一起使用，对每个变量能形成 J 图和 K 图。

对触发器 D 图而言，例如：分解成 J_D 和 K_D 图。每个图的现态是图坐标，而输入是控制变量值。每次跳变的状态都能很容易地从次态图中读出(图 10-2)。对于图 $D_{次态}$，触发器 D 当前值是 0，为顶上的两行，而 1 为底下的两行。考虑到这种情形，看第 2 行，第 3 列表示 0→1 跳变。这样在图 10-3，在 J_D 图中输入在 0111 是 1，而在 K_D 图中的为 X，如表 10.7(b)所示。

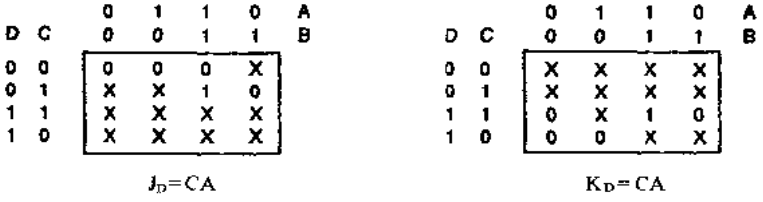


图 10-3

注意:对于 D 级,需要 1 个单个的 2 输入门代替两个 2 输入门和两个 3 输入门,余下的 J 和 K 图和相应的逻辑在题 10.4 中被论述。

虽然在本例中,用 JK 实现比用 D 触发器设计得到的组合逻辑简单,但不总是这种情况,设计者必须研究门数是否为一个重要的设计因素。

10.4　可编程逻辑器件的设计

图 10-1 所示是一个标准的 PAL 器件结构或是在第 9 章中描述的 PLA 器件。现态 / 次态表定义了组合逻辑的输出,其被寄存在触发器中,并构成电路的状态。如已经看到的那样,现态和次态之间的关系,可以用由真值表得到的布尔方程表示。烧片子的软件经常是直接接受这些代数方程。例如在例 10.1 中把计数器设计在 PAL 16R6 上就是这样考虑的。

例 10.4　PALIC 格雷(Gray)码计数器设计计数顺序和相关现态/次态真值表。(分别为表 10.1 和 10.2(b))为方便起见重复如下。

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

输入(现态)				输出(次态)			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

重复计数顺序　　对应的真值表

PALASM 语言的布尔方程直接由真值表得到并显示如下

```
D := /D*C*/B*/A + D*/C*/B*A + D*/C*B*/A + D*/C*B*A
     + D*C*/B*/A + D*C*/B*A + D*C*B*/A + D*C*B*A
C := /D*/C*B*/A + / D*C*/B*/A + / D*C*/ B*A + / D*C*B*/A
     + /D*C*B*A + D*C*/ B*/A + D*C*/B*A + D*C*B*A
B := /D*/C*/B*A + / D*/ C*B*/A + /D*/C*B*A + / D*C*B*/A
     + D*/C*B*/A + D*C*/B*A + D*C*B*/A + D*C*B*A
A := /D*/C*/B*/A + / D*/C*/B*A + /D*C*B*/A + /D*C*B*A
     + D*/C*B*/A + D*/C*B*A + D*C*/B*/A + D*C*/B*A
```

当这些方程被输入时,将使被烧的芯片产生所希望的连接,在 PAL 16R6 内连接成期望的逻辑电路。因此得到所希望的计数器。

10.5　ASM 表

在例 10.2 中，外部正/反变量被用于状态机上，我们看到所加的输入使设计状态表的长度增长一倍。如有大量的外部输入变量，状态表就增长的非常长，并且常常变得很庞大。这种情况下，设计者通常利用算术状态机表(algorithmic state machine (ASM)chart)。这种表，有意构成上述类似于计算机程序的流程图，使用了相对比较少的符号，逻辑上互相连接，表示状态机从一个状态到下一个状态的进程。这对系统的预期动作提供了一步一步的详细描述。

ASM 表符号

方框图用来表示指定的系统状态。常常在矩形框旁边标出助记符为状态命令，在矩形框内部列出的是当状态机在特定状态时可能产生的输出。这些输出叫无条件输出。状态机严格地按照指定状态存在一个时钟周期。

图 10-4 所示矩形框，表示状态 AA，而系统是在此状态中，它将产生输出 OUT1 和 XX。如果状态机以预定方式一个状态一个状态的进行，与任何输入无关。状态用直线段连接如图 10-5 所示。

在这种情况下，状态 J 存在一个时钟周期之后，当下一个时钟脉冲到达时，状态机的状态移到状态 K。注意这种情况下输出的变化。

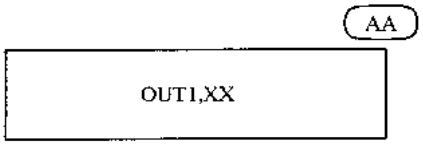


图 10-4

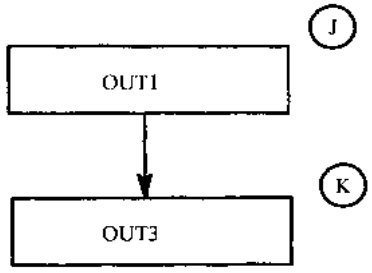


图 10-5

菱形或扩展菱形用来表示条件分支，次态假设决定于一个或多个输入变量的值。布尔函数与这些表示在扩展菱形内部变量有关，而分支路径和相应的函数值一起被标出。菱形本身叫决策框并且不会被分离标识出来。

在 ASM 表中线段表示在图 10-6 中，状态机在状态 J，将在下一个时钟脉冲进入状态 K 还是状态 L，决定于布尔函数 A + B 是(TRUE)真还是(FALSE)假。处于状态 J 的状态机在时钟周期内做出这个决策是很重要的。从这个意义上看，决策框属于被输入的状态。

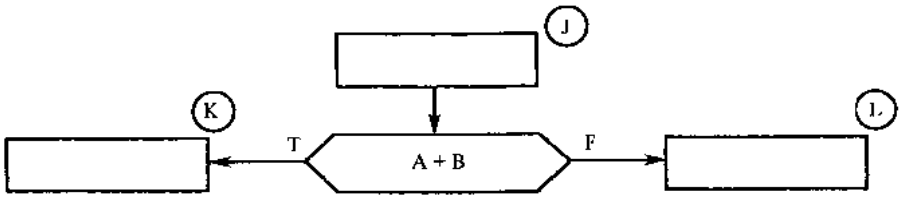


图 10-6

椭圆形用来表示条件输出，不只取决于状态，而且也与一个或多个输入的状态有关。它与决策框的退出有关，因此叫条件输出框。

图 10-7 所示表 1 状态机在状态 J 开始，在此时钟周期内，布尔函数 A(B + C)被估计出。在时钟周期内无条件地产生输出 OUT1，在此时 OUT2 或 OUT3 也将产生取决于布尔函数的

估值。在图 10-7 中，OUT2 有条件地产生状态 J，而无条件地产生状态 K。注意：决策框如何确定输出产生，在给定状态下同时也确定了次态。还要注意：在图中实际影响这些决策产生的只是那些输入。在改变的状态被确定之前没有必要对所有输入求值。

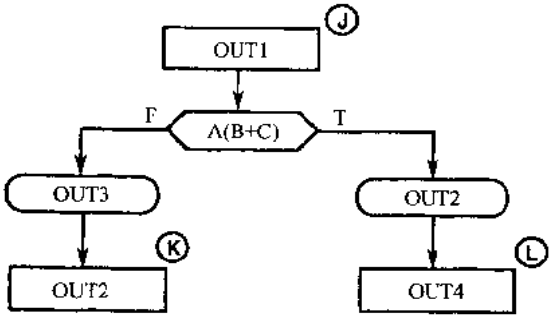


图 10-7

ASM 表结构

按照系统性能的进程如何一步一步地开发 ASM 表，借助例子说明也许是最好的。

例 10.5　增量编码器逻辑。

增量编码器广泛地用于控制系统的应用，它被认为是例 1.11 中描述的典型的双踪格雷码轴向编码器。输出信号产生两个方波形式。其中之一是相对另一个延迟 1/4 周期，如图 10-8 所示。已表明的增加和减小方向称格雷码计数方向为顺时针或逆时针方向旋转的轴向编码器。

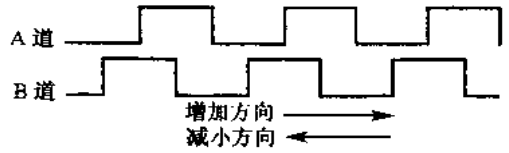


图 10-8　编码器输出波形

希望在二者在任一跟踪输出发生的瞬时产生一个脉冲。如果编码器以增量方向运动，脉冲就在 PU 输出，或编码器按减小方向旋转，则脉冲输出在 PD 端。这样在每个输出周期(见图 10-9)就应该有 4 个脉冲。



图 10-9　例 10.5 框图

设计 ASM 构成表是最容易的，方法是使每个提出的状态和已知的物理量相等。最终的表如图 10-10 所示。

考虑其结构，初始状态(标出 00)是任意选择，并且代表物理情形。两个编码器跟踪全输出 0s。只要在两者的跟踪上没有变化发生，就无期望的输出。并且系统保持在同一状态。当输入改变时，就改变状态。假设现在跟踪 B 变为 1，参见图 10-8 所示波形，运动必定是增加方向，PU 就产生脉冲输出。这样，次态对应的物理状态，跟踪 A 是 0，而跟踪 B 是 1(状态 1)。如果跟踪 A 变成 1，PD 脉冲就产生了。这样次态对应于物理情形便跟踪 A 产生一个 1，跟踪 B

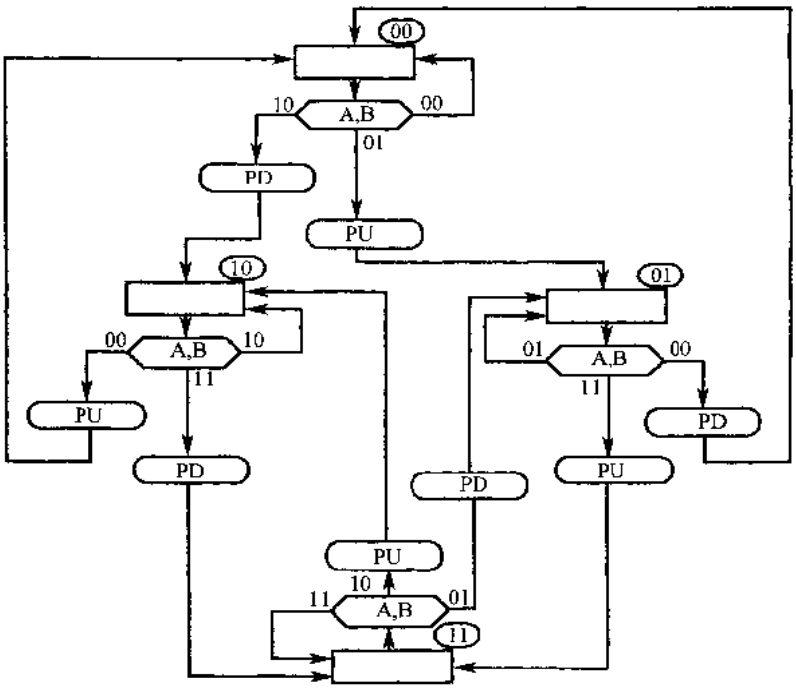


图 10-10

产生一个 0(状态 2)。注意:编码器的结构不允许跟踪 A 和 B 同时变为 1,在设计 ASM 表时用最上面的决策框确定跟踪的输出缺少 11,该框检查跟踪输出,并且使状态机直接到达次态。

事实上输出只出现在条件输出符号(椭圆)指示系统,其产生输出作为从一个状态到另一个状态变化的结果。这些条件输出只存在于决定 A 和 B 状态及下一个时钟脉冲之间的时钟周期的部分。因为这个原因,输入常被同步(即用状态机时钟同步 D 触发器),用非常窄的输出脉冲得到不同步输入的结果,可能是所含能量不足以触发任何减法电路。

状态 1 和 2 的跳变是受输入 A 和 B 的控制,同 A 和 B 的输入从 0 态变化方式一样,由于物理情况在某种意义上是有限的,所以 ASM 表与实际情况相符。

时钟脉冲定时是关键,输出脉冲发生在编码器变化的指定时刻。事实上,他们的发生跟在第 1 个时钟脉冲跳变的时刻,如果延时是最小的,和编码器的脉冲宽度相比时钟脉冲周期应该非常小。另一方面,PU 和 PD 的脉冲宽度等于一个时钟周期,对这些脉冲通常要求最小宽度。

10.6　从 ASM 表设计:用最少的 F.F 实现布尔方程

如果能构成一个适当的 ASM 表,就可能直截了当地确定它的逻辑图,图 10-10 的 ASM 表将用来说明这个过程。

第 1 步,产生状态转换表;

(a)考虑矩形框的状态数。在此情况下有 4 种状态。

(b)确定表示所有可能状态需要的触发器数,并把他们的输出表示在 ASM 表中。2 个触发 器可以表示 4 个状态,3 个触发器表示 8 个状态,等等。在图 10-10 的例子中,要求 2 个触发器,表示他们的输出 Q 分别为 X 和 Y,对状态 0,XY=0,状态 1 要求 XY=1,等等。状态转换表的第 1 列是用名字和触发器表示(表 10.8)列出的状态。

(c)列出目标状态表。在第 2 列中列出每一种可能的目标状态,并且使给定的输入状态都能达到。

(d)列出条件表。在第 3 列中列出任何条件,这些条件与每一状态到一状态的转换相联系。

在检测中，完成的状态转换表只是一种逻辑上的组织方向表，起一个等效的路径图的作用。

第 2 步，求控制器设计方程；

(a) 阅读状态转换表的第 2 列，并且注意当触发器在次态取值为 1 的情况(即下一个时钟脉冲之后)。

(b) 从第 3 列确定条件。例如第 3 行告诉我们，如果现态是 0，并且 AB′是真，则次态 X 应该是 0。

(c) 条件表示。令次态 X 或次态 Y 为 1，表示出现态和输入，得到布尔方程组，用同样的方法，逻辑方程也可以从真值表写出来，对所有的状态到状态的跳变一个一个写出其所要求的条件。这个逻辑被称作控制器方程。

从真值表 10.8 得到控制器方程为

$X_{次态}$ = "0"AB′ + "1"AB + "2"A + "3"A

$Y_{次态}$ = "0"A′B + "1"B + "2"AB + "3"B

(d) 编码的状态名和触发器输出值。得到的控制器设计方程所表示的逻辑函数是当前的状态触发器：

$$X_{次态} = X'Y'AB' + X'YAB + XY'A + XYA \tag{10.1}$$

$$Y_{次态} = X'Y'A'B + X'YB + XY'AB + XYB \tag{10.2}$$

表 10.8　状态转换表

现态 XY	次态 XY	条件
0　00	0　00	A′B′
	1　01	A′B
	2　10	AB′
1　01	0　00	A′B′
	1　01	A′B
	3　11	AB
2　10	0　00	A′B′
	2　10	AB′
	3　11	AB
3　11	1　01	A′B
	2　10	AB′
	3　11	AB

如果使用 D 触发器，就接到 D 输入端；如果使用 JK 触发器，方程要分成 J 和 K 两部分如第 10.3 讨论的那样。

第 3 步，求输出设计方程；

(a) 在 ASM 表中，确定好每一个参考输出端，并决定其条件是否任意或与参考条件相关。

(b) 用布尔或关系把所有参数连接起来。

在图 10-10 中，变量 PU 在 4 处发生，全部是条件输出，当系统是在状态 0 并且当 A′B 是真时，最上面的 PU 有输出，其他的例子类似于定义和导出下面的布尔方程：

PU = "0"A′B + "1"AB + "2"AB′ + "3"AB′

编码名状态送到触发器值之后，就得

$$PU = X'Y'A'B + X'YAB + XY'AB' + XYA'B' \tag{10.3}$$

同样对 PD 方程的结果是

$$PD = \text{“0”}AB' + \text{“1”}A'B' + \text{“2”}AB + \text{“3”}A'B$$
$$= X'Y'AB' + X'YA'B + XY'AB + XYA'B \quad (10.4)$$

方程(10.1～10.4)为状态机构成的一个完全的设计方程组，设计者能以任何希望的形式实现。如果使用小规模集成电路，方程可用图来简化，但更喜欢使用可编程逻辑器件。它只是一个把布尔方程翻译成设计软件需要的语言问题，问题 10.9 说明了某些设计者优先选用同步状态机的方法。

10.7　从 ASM 表设计：单状态(one-hot)控制器的实现

对于小的控制器，有些设计者宁愿有多少状态就用多少个状态触发器，每个触发器表示一个状态，当处于某个状态时，对应触发器只有 1 个是工作的，即它的输出为有效。在控制器中使用译码逻辑比使用最小触发器实现还简单。象前面所说的那样，设计一个单状态控制器，从 ASM 表写出状态转换表开始，然而这时的信息要列出导致给定现态的前面所有的状态。

控制器设计方程

参考表 10.9 的顶框，从图 10-10 确认，如果输入 A 和 B 都是“无效”，触发器的 D 输入“0”应该是 1，不考虑以前的状态。继续观察，从表上看到对给定的现态，所有输入条件都是同样的，与以前的任何状态无关，这样 4 个设计方程是“0” = A′B′，“1” = A′B，“2” = AB′，和“3” = AB。

输出方程

输出方程类似于第 10.6 节得到结果。

$$PU = \text{“0”}A'B + \text{“1”}AB + \text{“2”}AB' + \text{“3”}A'B'$$
$$PD = \text{“0”}AB' + \text{“1”}A'B' + \text{“2”}AB + \text{“3”}A'B$$

对于完整的单状态逻辑控制器的解决方案如图 10-11 所示。

表 10.9

前态	现态	条件
0	0	
1		
2		
3		无关
0	1	
1		
3		
2		无关
0	2	
2		
3		
1		无关
3	3	
1		
2		
0		无关

10.8　从 ASM 表设计：状态表输入到可编程器件

当状态机在可编程器件上实现时，如第 10.2 节指出的那样，可以使用布尔输入，然而很多设计者宁愿使用软件技术来定义所有状态和 ASM 表的信息，这称作状态表输入形成结构程序设计。

例 10.6　考虑例 10.5 增量编码器的设计要求，根据状态表输入，使用 INTEL ipLSII 软

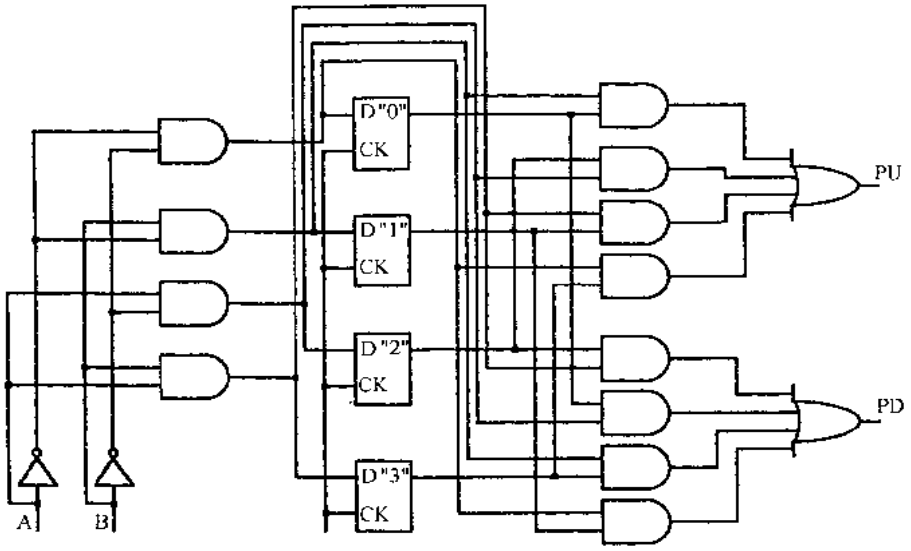


图 10-11　单状态控制器的实现

件。状态被定义为借助矩阵用状态触发器表示，然后，在时序上对每个状态、激励和输出都表示出来。IF-THEN 或 CASE 语句用于条件激励、无条件激励，当前状态都被列出来。

输入	解释
HEADER	不包含在此例中
MACHINE:Counter Example	系统名
CLOCK:CLK	时钟名
STATES:[FF1　FF2]	
ZERO[0　0]	状态条件
ONE[0　1]	
TWO[1　0]	
THREE[1　1]	
ZERO:	
IF A′ * B′ THEN ZERO	条件激励到次态
IF A′ * B THEN ONE TWO	无效激励。因为两个编码器跟踪输出决不同时改变，从 00 跳到 11 不会发生。这样，状态只有保持的可能性
ASSERT:	
IF A′*B THEN PU	条件输出
IF A*B′ THEN PD	
ONE:	
CASE	选择 IF-THEN 形式条件激励到次态
A′*B′:ZERO	
A*B:THREE	
ENDCASE	
ONE	无效激励(AB=10 不会发生)

续表

输入	解释
ASSERT：	
IF A′*B′THEN PD	条件输出
IF A*B THEN PU	
TWO：	
CASE	
A′*B′:ZERO	条件激励到次态
A*B:THREE	
ENDCASE	
TWO	无效激励(状态 01 不会发生)
ASSERT：	
IF A′*B′ THEN PU	条件输出
IF A*B THEN PD	
THREE：	
CASE	
A′*B:ONE	条件激励到次态
A*B′:TWO	
ENDCASE	
THREE	无效激励(状态 00 不会发生)
ASSERT：	
IF A′*B THEN PD	条件输出
IF A*B′ THEN PU	
END $	

在状态说明里，输出接着被列出，后跟着 ASSERT 这个词。条件输出正如出口一样，能通过 IF(条件)，THEN(输出)或者 CASE 语句准确地列出。

当状态表输入程序被完成，它被输入到个人计算机中。iPLSII 软件翻译状态表成为一个适当的布尔方程组。如果希望的话一般能由设计者检查。然后就开始了器件的程序设计。如果程序设计已经使用，程序设计操作已被更多的用户友好地使用，因为它允许设计者直接输入来自他的设计工具 ASM 表程序设计信息。

10.9　状态机的时钟

同步状态机的心律时钟脉冲，假设同时出现在电路中的所有点上。在很多实际情况，不可能保持精密地同步：因为各种不同时钟路径有不同的传播延迟。设想两个重叠时钟信号出现有时间差的电路上，被称作时钟偏斜，在高速电路中，这可能是主要的错误源。

例 10.7　为了说明是时钟偏斜的影响，将图 7-18 重画如图 10-12 所示。

为方便起见，第 1 个和最后 1 个触发器分别标上了 A 和 B。假设在“同一”时钟脉冲到达触发器 A 之后，时钟脉冲在某时刻到达引起触发器 B 的 D 输入出现任何变化，重叠时钟脉冲到达这个触发器之前，考虑事件发生的顺序：

1. 时钟脉冲到达触发器 A；

2. 传播延迟之后：触发器 A 改变引起与门输入的逻辑变化；

3. 另一个时钟传播延迟之后，一个或多个与门的变化引起或门输入的改变；

4. 再一次传播延迟之后，或门上的变化引起另一个触发器(在这种情况为 B)D 输入的改变；

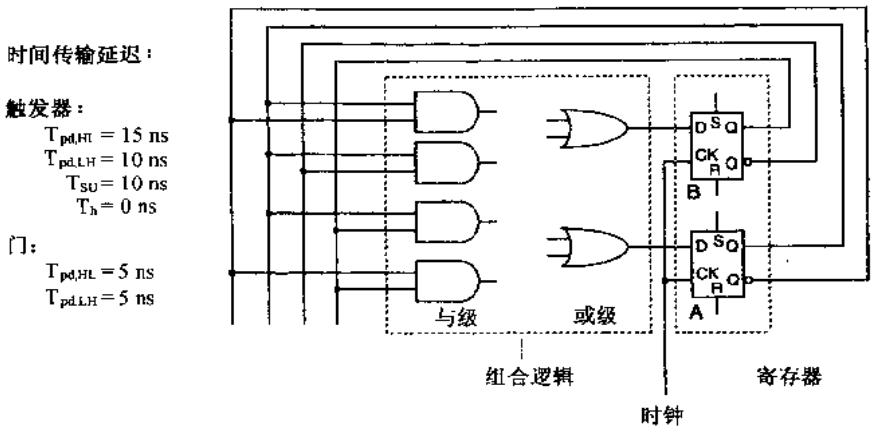


图 10 12

5.最小的建立时间之后,B 触发器将为不正确的信息。

最小时间需要 5 步完成,表示系统所能允许的最大延迟(时钟偏斜)。使用在图中标好的参数,系统所允许的最大时钟偏斜误差就能计算出 10 + 5 + 5 + 10 = 30ns。

注意:在例 7.5 中计算最大可允许时钟脉冲频率,要求知道最大时钟传输延迟时间,而计算最大时钟偏斜要使用最小延迟。

触发器 B(第 4 步)D 输入的改变引起建立时间违反触发器规律也是值得注意的。如果在问题的设计当中,稳定性失败,最大可接受的偏斜就要减小前 4 步的时间延迟,在这个例子中是 20ns。

高速数字系统的设计者必须考虑时钟偏斜,并且常常取很大的长度使时钟路径相等,或减小到最小传输延迟。当制造者已经指定定时参数最大值时,在很多情况下也给出最小值。

10.10　状态机中的初始化与锁定

在设备接通以后初始化状态机成为已知状态,通常有着非常重要的意义,往往使其状态设定为所有触发器全为 0,清零或复位命令可用于初始化,如图 10-13 所示。

在单状态控制器(第 10.7 节)实现中,全零情况是不能发生的。在这种情形下,设计者常常采取使触发器交换 Q 和 Q′的状态的方法。Q′输出接到逻辑中的所有点。Q 一般正常连接。当清零线加上脉冲时系统就适当初始化为单状态。注意:加于触发器 D 输入端逻辑非的信号也是必要的,以至当 1 为希望值时,产生 0,这个变化如图 10-14 所示。

如果状态机不被初始化,当设备加电时,能随机地进入假设"无关最小项"或者未被指定的状态;可能进入所加时钟脉冲数不可能出现的状态。这样的情况称作锁定。并且在题 8.11 中说明。对锁定问题最有效的解是,通过提供自适应的或外加初始化信号来避免锁定。

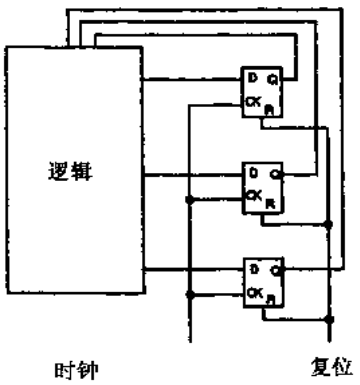


图 10-13

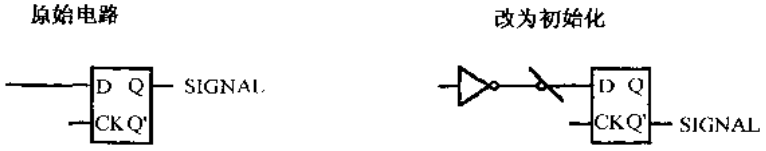


图 10-14

习题解答

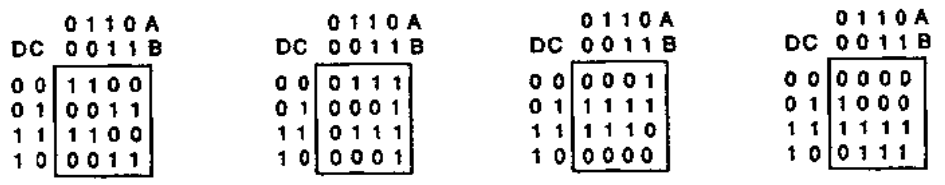
10.1　使用小规模集成电路，实现例 10.1 的格雷码计数器状态机，写出估计要用的芯片数。

解　设计逻辑的真值表已在前面推导出(见表 10.2)并重新表示如下：

现态				次态			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

利用与变量无关的坐标设定为现态，对 4 变量中的每个次态用 K 图描述。如例 10.3 描述的那样。

通过利用 $X \oplus Y = XY' + X'Y$ 和 $(X \oplus Y)' = XY + X'Y'$ 公式可以得到如图 10-15 所示 d_A 的简化布尔表达式。异-或形式可从图的检查板结构中预示出来(参考第 3.5 节)。



$d_A = (B \oplus C \oplus D)'$　　$d_B = BA' + DCA + D'C'A$　　$d_C = CB' + CA + D'BA'$　　$d_D = CB'A' + DA + DB$

图 10-15

SSI 芯片数：2 -异或门　　(1 片)

五-2 输入与门　(2 片)

四－3 输入与门 (2 片)

一 3 输入或门 (1 片)

所有的原变量和反变量(4 个反变量)直接出现，对异或要求 1 个反相器。5 个反相器一起要求至少 1 个芯片，总共为 7 个。注意：把两个空闲的 3 输入的与门接成两输入与门，有可能节省 1 个芯片，完整的计数器逻辑电路如图 10-16 所示。

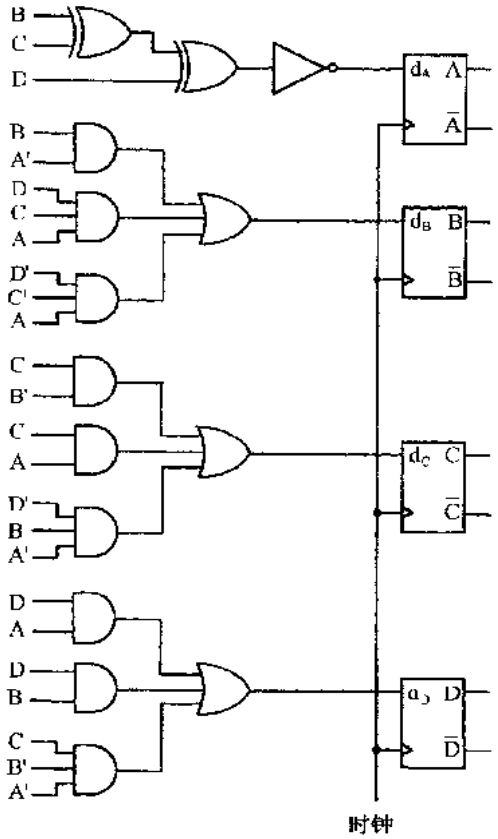


图 10-16　解答问题 10.1

10.2　表 10.10 所列为一个余 3 计数器的计数顺序，计数器计数是从二进制表示的十进制 3 到十进制 12，然后再重复。使用 ROM 和 D 触发器，设计此计数器为寄存现态的状态机。

表 10.10

A	B	C	D
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

重复

表 10.11

十进制 ROM 地址	现态 A B C D	次态 A B C D
3	0 0 1 1	0 1 0 0
4	0 1 0 0	0 1 0 1
5	0 1 0 1	0 1 1 0
6	0 1 1 0	0 1 1 1
7	0 1 1 1	1 0 0 0
8	1 0 0 0	1 0 0 1
9	1 0 0 1	1 0 1 0
10	1 0 1 0	1 0 1 1
11	1 0 1 1	1 1 0 0
12	1 1 0 0	0 0 1 1

解 用状态表(见表 10.11)作为 ROM 编程的基础,其结果状态机如图 10-17 所示。

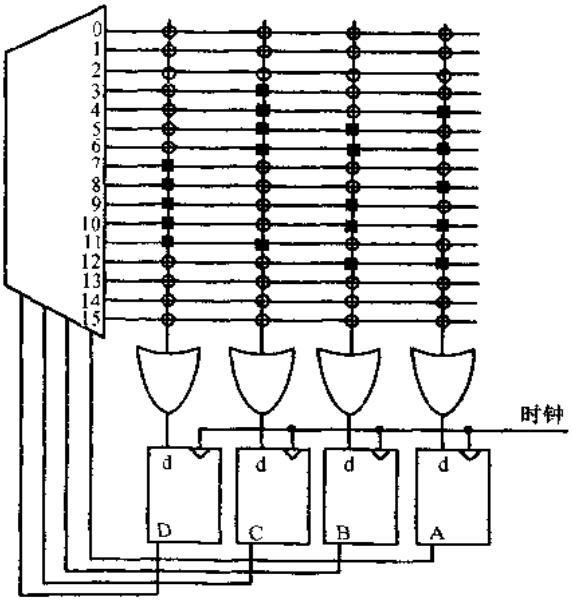


图 10-17

10.3 参考问题 10.12 使用 PAL 16R6 IC 作为硬件,如何组建成一个余 3 的 4 位可逆计数器。

解 一个好的设计方案是用触发器输出表示状态,应用现态/次态表设计。这样的表在第 2 章(问题 2.33)已讨论过。

在 PAL 16R6 有 6 个触发器作为计数状态只要求 4 个,然而,组合逻辑的每 1 个输出是含 10 个与项,并不适合 PAL 器件,因为驱动每个触发器的只有 8 个与门,所以有必要进行化简。在问题3.28 中 5 变量 K 图化简成 2 个变量;An 和 Cn 简化成 6 个与项的函数,剩下的两个变量容易化简成 8 项或更少。

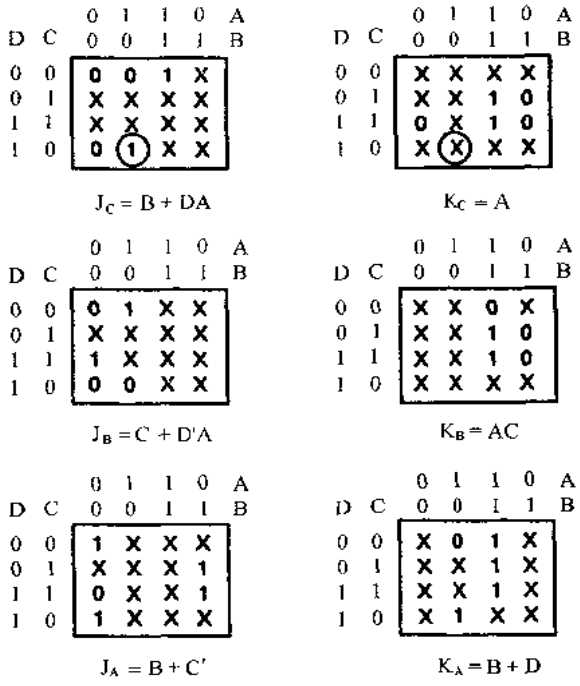


图 10-18

下面的简化过程,使计数器能用带有剩余空间的 PAL16R6 实现,一组设计方程如下

$$A_n = (U/D)'BCD + AD + AC + (U/D)'AB' + (U/D)AB + (U/D)A'B'$$

$$B_n = (U/D)'A'B'CD + (U/D)'A'BC' + (U/D)'A'BCD' + (U/D)'AB'CD + (U/D)A'B'CD + (U/D)A'BC'D + (U/D)A'BC + (U/D)'AB'C'D'$$

$$C_n = (U/D)'CD' + (U/D)'C'D + (U/D)C'D' + (U/D)'AB + (U/D)'ACD + (U/D)BCD$$

$$D_n = D'$$

10.4　完成例 10.3 的 JK 表并画出电路结构图。

解　所要求的图参考了图 10-2 和表 10.7。第 1 步，在 D 图中所有“无关最小项”输入全被转换成 JK 图输入对。典型的 JK 输入如下：考虑图 10-2 中的 Cnext 图的第 4 行第 2 列。它告诉我们，如果现态是 1001，C 的次态值将是 1。这样，如果 Jc = 1，Kc = X，表 10.7 给出的情况就会发生，C 将从 0 变成 1。这些输入被圈入图 10-18 中的第 1 对。电路图如图 10-19 所示。

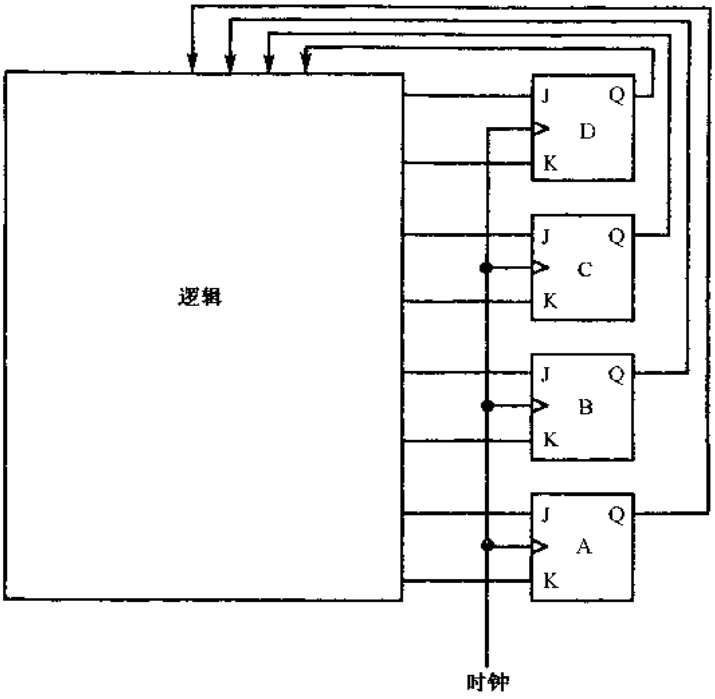


图 10-19

10.5　对表 10.12 指定的计数器，要求用触发实现状态机的逻辑电路。

解　列出状态表(表 10.13)，注意有 6 个“无关最小项”状态：0010，0100，0101，1010，1011，和 1101。对每个变量画出 D 图，然后使用表 10.7 的转换关系扩展到相应的 JK 图，概述如下：

如果次态是 1：

当 Q = 1 时，则 J = X 并且 K = 0；

当 Q = 0 时，则 J = 1 并且 K = X。

如果次态是 0：

当 Q = 1 时，则 J = X 并且 K = 1；

当 Q = 0 时，则 J = 0 并且 K = X。

在得到的一组图(图 10-20)中，指定的“无关最小项”用黑体标出，以区别和 JK 转换过程有关的“无关最小项”。

表 10.12

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	1	0	0
1	1	1	0
1	1	1	1

重复

表 10.13

现态				次态			
D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

10.6　考虑表 10.14 给出的单向计数器，假设两个方向和触发器的互补输出均可得到，推导出使用的系统逻辑方程。并用 JK 触发器实现上述要求。并从硅片使用和最大时钟速率的观点比较两个设计方案。

表 10.14

A	B	C
0	0	0
0	1	0
0	1	1
1	0	1
1	1	0
0	0	1
1	0	0
1	1	1

重复

解　☞ 利用下面的指标：

1. D-触发器和 JK 触发器每种都使用 10 片，两种都有 20 个时间单元的建立时间，5 个时间单元的保持时间，和 25 个时间单元的传输延迟时间。

2. 门有 10 个时间单元传播延迟，并使用 4 个硅单元，每个输入加 0.5 个单元。（二输入 4 或门）。

有关 D 和 JK 图设计分别表示在图 10-21 和图 10-22 中。

D 触发器设计

使用硅片：

A_N　15 硅片$(4*3+6*0.5)$

B_N　10(只对 A′C′，因为 AB′已得到)

C_N　15

D 触发器 $\frac{30}{70\text{ 个硅单元}}(3\times10)$

定时：

两级逻辑：20 个时间单元

触发器上升时间：20

传输时间：$\frac{25}{65}$个时间单元

JK 触发器设计

使用硅片：

$J_A K_A$　0 硅片(直接连接)

J_B, K_B　10　(每个输入端 1 个 2 输入或门)

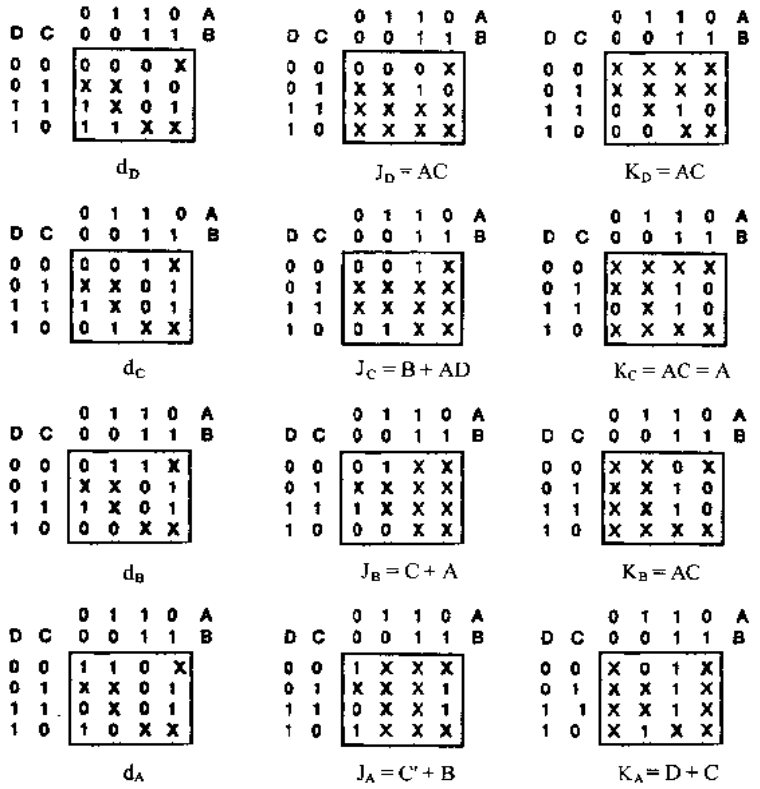


图 10-20　题 10.5 的设计

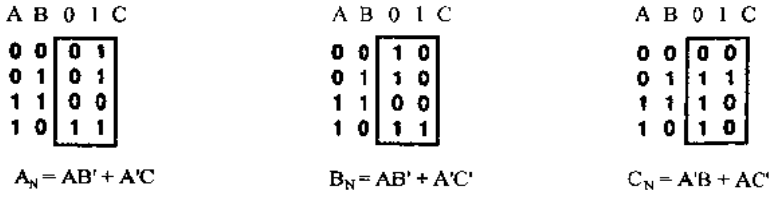


图 10-21　题 10.6 D 触发器设计

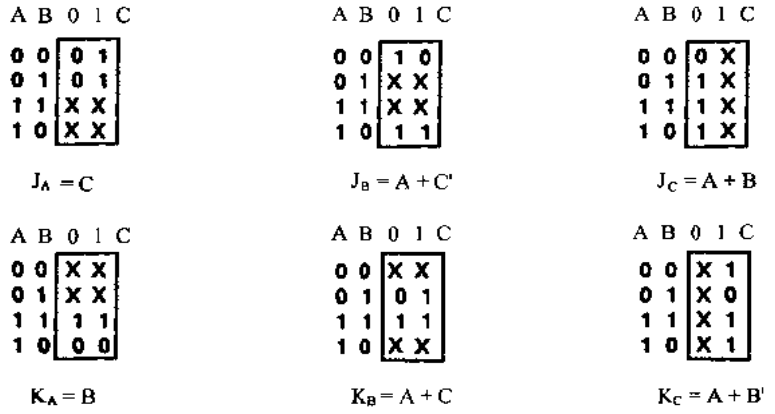


图 10-22　题 10.6 JK 触发器设计

J_C, K_C　10　（每个输入端 1 个 2 输入或门）

JK　触发器　$\frac{30}{50\text{ 硅单元}}$（比 D 设计少用 29 %）

定时：

门延迟：10 个时间单元

触发器上升时间：20

传输时间：$\frac{25}{55}$个时间单元(比 D 设计快 15%)

10.7　使用多路选择器实现图 10-10 的系统逻辑电路。要求用 D 触发器的数量最少。

解　一个有效的策略是使用现态输出作为多路选择器的输入，如果每个多路选择器输出接到状态触发器的输入，那末，这个问题就变成要确保与现态有关多路选择器的数据输入给触发器提供适当次态值。

利用第 10.6 节状态转换表。

现态 XY	次态 XY	条件
0 0 0	0 0 0	A′B′
	1 0 1	A′B
	2 1 0	A B′
1 0 1	0 0 0	A′B′
	1 0 1	A′B
	3 1 1	A B
2 1 0	0 0 0	A′B′
	2 1 0	A B′
	3 1 1	A B
3 1 1	0 0 1	A′B
	2 1 0	A B′
	3 1 1	A B

从表上知道如果系统在状态 0，状态被清零，只要 AB′有效，触发器的次态 X 就是 1。这样，逻辑实现 AB′被接到多路选择器的输入 0，起到触发器作用。同样，对于 AB 逻辑接到数据输入 1 上，(由状态输出选择)。在数据输入的情况下，其逻辑是 AB′ + AB，能化简成 A。对 X 和 Y 多路选择器，表示在图 10-23 中。输出逻辑从第 10.6 节中的式 10.3 和式 10.4 产生的。

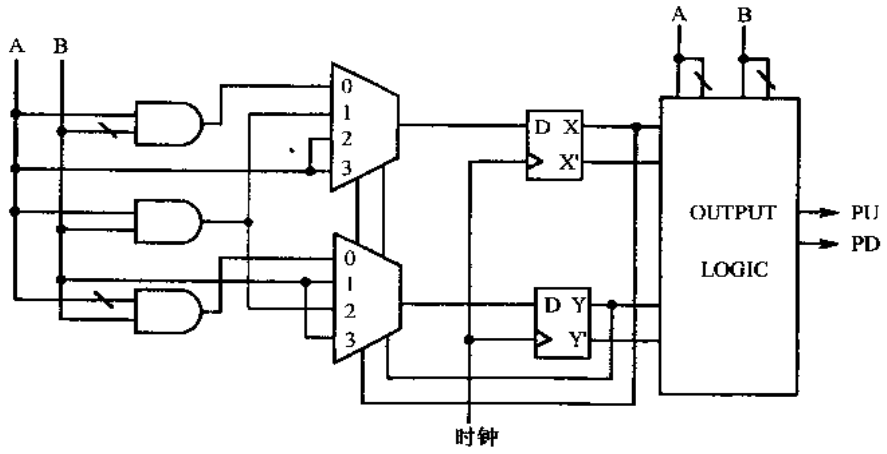


图 10-23　用多路选择器实现控制器

如果我们对使用 SSI 和 MSI 芯片设计增量编码器有兴趣的话，用这些芯片实现一个单状态和用多路选择器芯片实现进行比较是有益的。(分别为 图 10.11 和 图 10.23)。

单状态实现：

输出逻辑　2 个 4 输入门(2 个/片)1 片

　　8 个 2 输入门(4 个/片)2 片

触发器　6 个 D 触发器(4 个/片)2 片

A,B 译码器　4 个 2 输入门(4 个/片)1 片

2 个反相器(6 个/片)$\frac{1\text{片}}{7\text{片}}$

多路选择器实现：

输出逻辑　10 个 4 输入门(2 个/片)5 片

控制器　2 个 2 输入 MUX(2 个/片)1 片

触发器　4 个 D 触发器(4 个/片)1 片

MUX 门　3 个 2 输入门(4 个/片)$\frac{1\text{片}}{8\text{片}}$

10.8　已知图 10-24 所示流程图，写出状态变量和输出的设计方程。假设本设计使用了最少数量的 D 触发器。

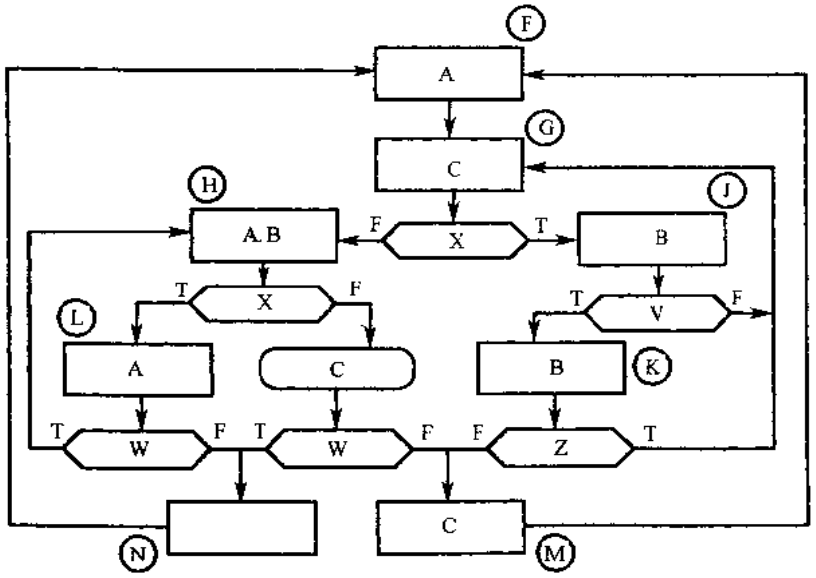


图 10-24

解　有 8 个状态，所以要求 3 个触发器，将其标为 R,S 和 T。如图示，状态已被表示成 F,G,H,J,K,L,M 和 N。

按照第 10.6 节每第 2 步，构成 1 个状态转换表(表 10.15)，R,S,T 状态是任意按字母顺序安排的。

表 10.15

现态 R S T	次态 R S T	条件
F　0 0 0	G　0 0 1	无条件
G　0 0 1	H　0 1 0	X′
	J　0 1 1	X
H　0 1 0	L　1 0 1	X
	M　1 1 0	X′W′
	N　1 1 1	X′W
J　0 1 1	K　1 0 0	V
	G　0 0 1	V′
K　1 0 0	G　0 0 1	Z
	M　1 1 0	Z′
L　1 0 1	N　1 1 1	W′
	H　0 1 0	W
M　1 1 0	F　0 0 0	无条件
N　1 1 1	F　0 0 0	无条件

接着，使用在第 3 步描述的方法写出控制方程。

$$d_R = H + JV + KZ' + LW' = R'ST' + R'STV + RS'T'Z' + RS'TW'$$

$$d_S = G + HX' + KZ' + L = R'S'T + R'ST'X' + RS'T'Z' + RS'T$$

$$d_T = F + GX + H(X + W) + JV' + KZ + LW' = R'S'T' + R'S'TX + R'ST'(X + W) + R'STV' + RS'T'Z + RS'TW'$$

注意：某些布尔化简已经使用，例如，在第 3 个方程中，H(X+W)项被简化成 H(X + X ' W)。

最后，输出方程可以直接从 ASM 表写出。例如，我们看到输出 A 产生后，当系统是在 F、H 或 L 状态时：

$$A = F + H + L = R'S'T' + R'ST' + RS'T$$

$$B = H + J + K = R'ST' + R'ST + RS'T'$$

$$C = G + M + HX' = R'S'T + RST' + R'ST'X'$$

10.9　设计一个 ASM 表，对应图 10-25 所示的同步状态机。

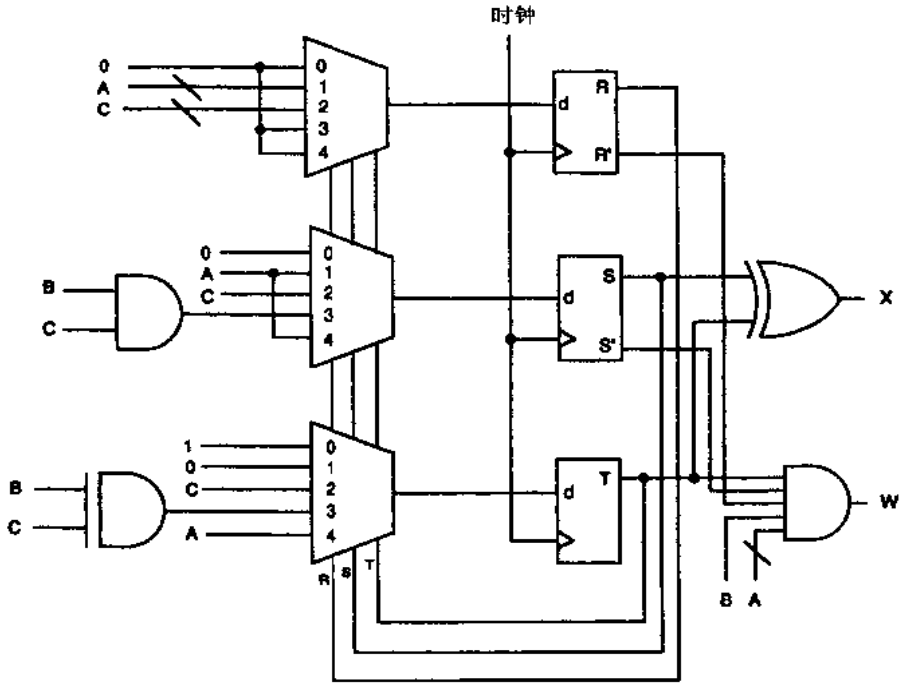


图 10-25

解　☞ 左下角的符号是一个等效门，用正逻辑，当两个输入相等时产生真输出。变量 B 和 C 等值时，用布尔形式表示成 F = B'C' + BC。

启动 RST = 000 并且由观察每个时钟脉冲寻址多路选择器，决定每个次态。

当状态是 000 时，清零，次态将是 001；

当状态是 001 时，D 输入是 A'A0，意味着次态就是 010(2)，如果 A = 1，并且 A = 0，次态就是 100(4)；

当状态是 010 时，d 输入是 C'CC，如果 C = 1，于是将得到状态 3；并且如果 C = 0 则为状态 4。

对于状态 3，D 输入分别是 $d_R = 0$，$d_S = BC$　$d_T = (B'C' + BC)$。借助一个小的真值表(表 10.16)容易得到各种可能的分析。

表 10.16

	B	C	R	S	T	状态
	0	0	0	0	1	"1"
BC + BC'	0	1	0	0	0	"0"
	1	0	0	0	0	"0"
	1	1	0	1	1	"3"

状态 4 产生的 d 输入为 0AA，如果 A＝1，次态就是 3；如果 A＝0，次态就是 0。

从上面的信息知道，我们现在可构成一个状态转换表，启动状态为 RST＝000(表 10.17)。

表 10.17

	现态 RST	次态 RST		条件
"0"	0 0 0	"1"	0 0 1	无条件
"1"	0 0 1	"2"	0 1 0	A
		"4"	1 0 0	A′
"2"	0 1 0	"3"	0 1 1	C
		"4"	1 0 0	C′
"3"	0 1 1	"3"	0 1 1	BC
		"1"	0 0 1	B′C′
		"0"	0 0 0	BC＋BC′
"4"	1 0 0	"0"	0 0 0	A′
		"3"	0 1 1	A

注意：如果 A 是假，从状态 1 到 4，忽略 B 值。

从图 10-25 所示，当状态为 001 或 010(1 或 2)时，我们看到输出 X 产生；但只有当 A 是假并且 B 是真时，输出 W 属于状态 001。输出可用布尔方程形式表示为：

$$W = R'S'TBA \quad [\text{状态“1” 和 } BA'(\text{条件输出})]$$

$$X = ST' + S'T[\text{状态“1” 或状态“2”}]$$

ASM 图表可画出，如图 10-26 所示。

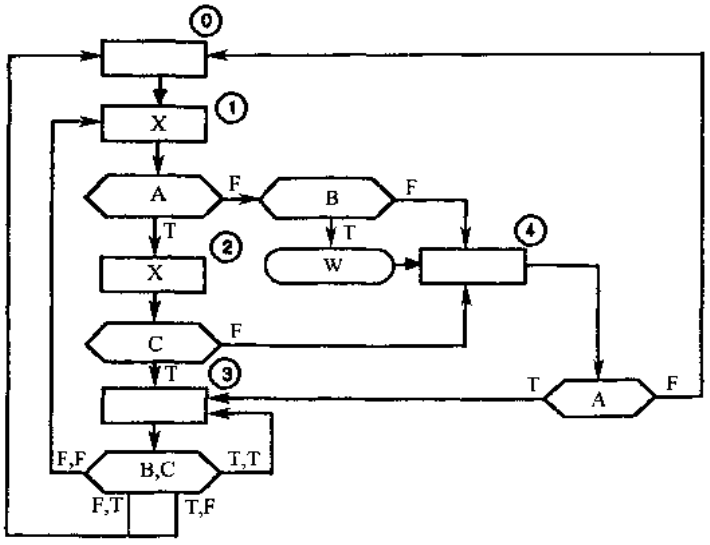


图 10-26

10.10　使用图 10-27 所示 ASM 表，实现一个反应式单状态控制器。

解　对于 6 个状态中的每一个，通过表返回决定前面的状态和它们每一个相关联的转化条件，这个分析的结果被组织和表现在表 10.18。

对每个现态可直接从表写出控制器方程：

完整的一个单状态控制器的设计表示在图 10-28 中，注意：初始化状态，包括触发器 L，在第 10-10 节中有描述。

10.11　参考问题 10-10，写出用最少的触发器设计的布尔方程。

解　3 个触发器要求编码成 5 个状态。称这 3 个触发器分别为 U，V 和 W，由跟踪图 10-27 的 ASM 表中的时序逻辑，或者通过转换前面的状态成为现态，而现态成为表 10.18 的次态，并重新安

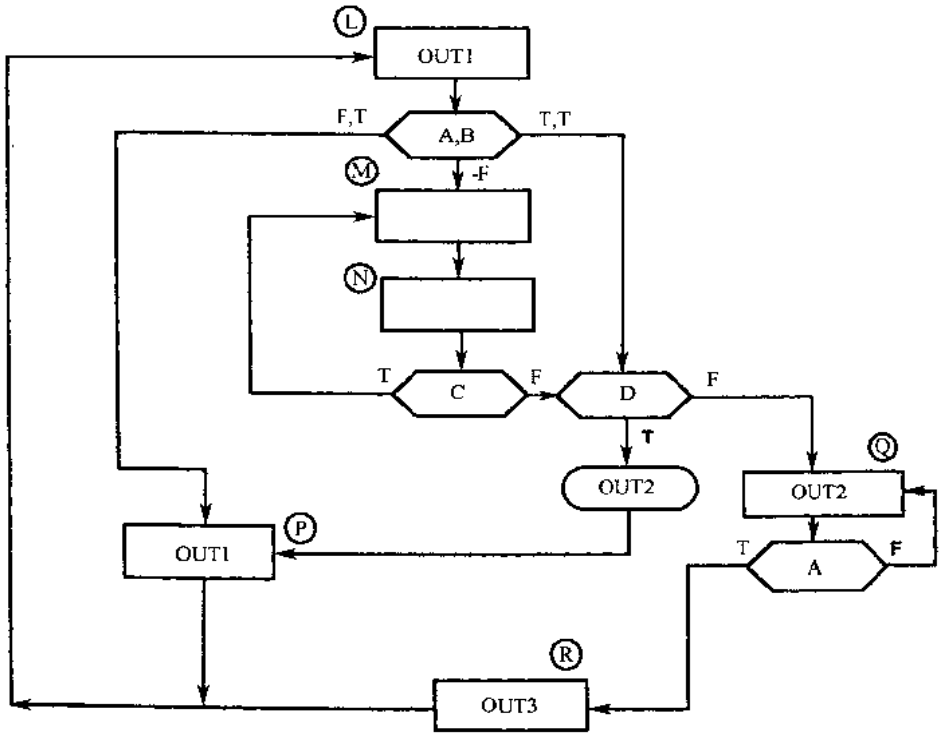


图 10-27　问题 10.10 的 Asm 表，"F"符号在决策框 A 的下出口处，B 表示若 B 为假，A 为"无关最小项"，即它或者真或者假。

表 10.18

前态	现态	条件
R P	L	无条件
L N	M	B′ C
M	N	无条件
L N	P	A′B + ABD C′D
L N Q	Q	A B D′ C′D′ A′
Q	R	A

排输入可得到状态转换表。结果表示在表 10.19。

对每个 D 触发器，全部是次态为逻辑 1 的条件。控制器的设计方程通过写在下面的式子得到，通过在适当的地方运用恒等式 A.13(附录 A)，下面的最终方程已经被化简。

$$D_U = LABD' + NC'O' + Q = U'V'W'ABD' + U'VW'C'D' + UV'W'$$

$$D_V = L(A'B + ABD) + M + NC'D = U'V'W'B(A' + D) + U'V'W + U'VW'C'D$$

$$D_W = L(B' + A'B + ABD) + N(C + C'D) + QA - L(B' + B(A' + AD)) - N(C + D) + QA$$
$$= U'V'W'(B' + A' + D) + U'VW'(C + D) + UV'W'A$$

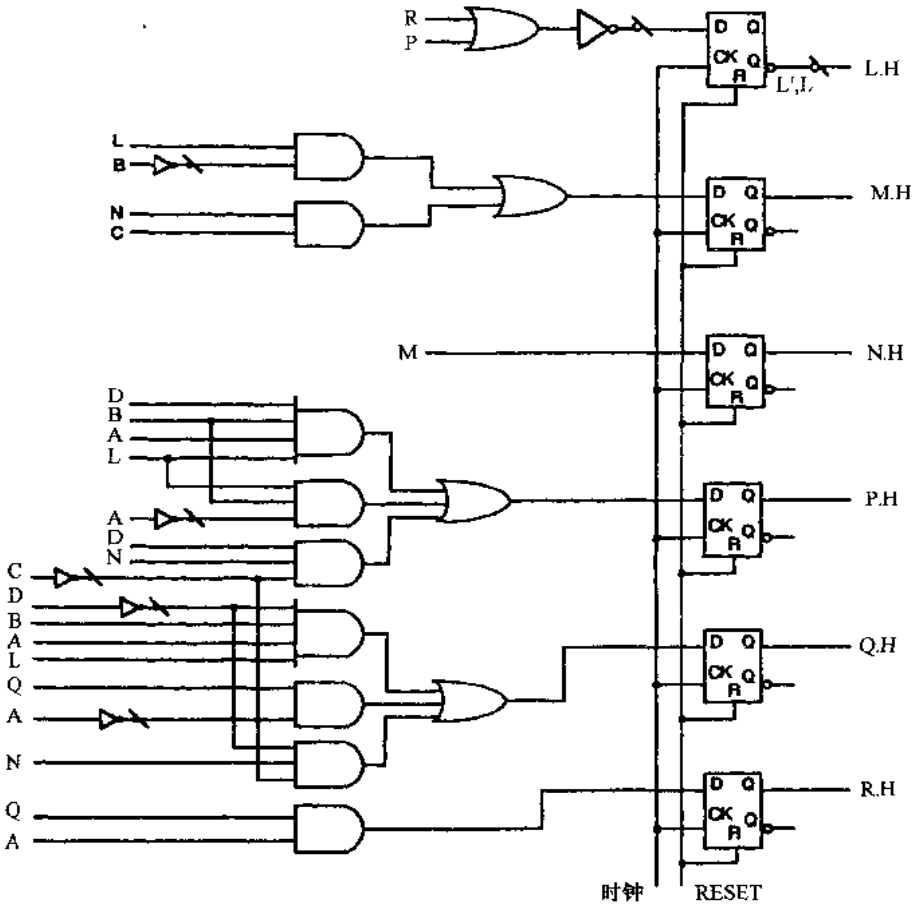


图 10-28

表 10.19

现态 UVW		次态 UVW		条件
L	0 0 0	M	0 0 1	B′
		Q	1 0 0	A B D′
		P	0 1 1	A′B + ABD
M	0 0 1	N	0 1 0	无条件
N	0 1 0	M	0 0 1	C
		P	0 1 1	C′D
		Q	1 0 0	C′D′
P	0 1 1	L	0 0 0	无条件
Q	1 0 0	Q	1 0 0	A′
		R	1 0 1	A
R	1 0 1	L	0 0 0	无条件

输出方程可以直接从 ASM 表写出来。

OUT1 = L + P = U′V′W′ + U′VW

OUT2 = Q + LABD + NC′D = UV′W′ + U′V′W′ABD + U′VW′C′D

OUT3 = R = UV′W

10.12　把问题 10.10 的解转换成为状态输入表。

解　状态表输入程序的基本部分由状态矩阵和每一个状态的条件的列而组成，象表 10.19 得

到的那样。

状态	$[Q_U$	Q_V	$Q_W]$
L	[0	0	0]
M	[0	0	1]
N	[0	1	0]
P	[0	1	1]
状态	$[Q_U$	Q_V	$Q_W]$
Q	[1	0	0]
R	[1	0	1]

```
L:
    IF B' THEN M
    IF A'*B+A*B'D THEN P
    IF A*B*D' THEN Q
    ASSERT:
        OUT1
        IF A'B*D THEN OUTPUT 2
M:
    N
N:
    IF C THEN M
    IF C'*D THEN P
    IF C'*D' THEN Q
    ASSERT:
        IF C'*D THEN OUT2
P:
    L
    ASSERT:
        OUT1
Q:
    IF A THEN R
    IF A' THEN Q
    ASSERT:
        OUT2
R:
    L
    ASSERT:
        OUT3
```

10.13 使用题 10.10 的 ASM 表和图 10-29 所示波形，画出所有状态和输出的时序图，假设在此之前是复位状态。

解 提示：输入跳变与时钟脉冲一致，表示这是一个同步状态机。当复位时，状态机为初始状态 L，如给定的那样。

参考图 10-27 的 AMS 表有如下的态序：

1. 因为第一个时钟脉冲之前 A 和 B 为 1，而 D=0，状态机将变成状态 Q。
2. 在脉冲 2，A=1 并且状态机转换成状态 R。
3. 下一个状态为无条件的 L。
4. A=1，B=1，D=1；状态为 P。
5. 无条件转换到 L。
6. A=0，B=1；状态为 P。
7. 无条件返回到 L。

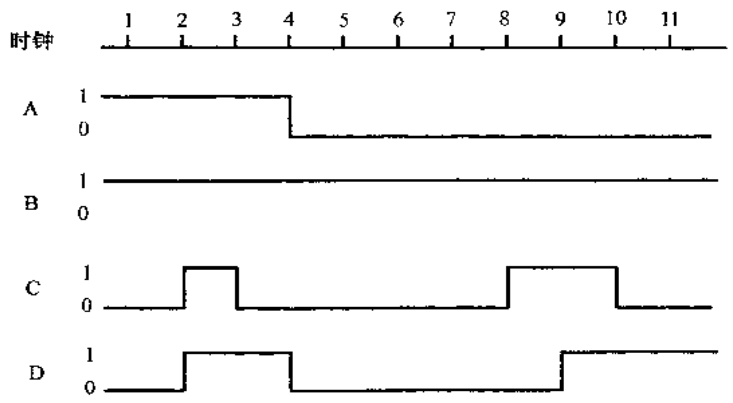


图 10-29

8. 因为 A 和 B 只决定后面的状态 L，它们将在后续 4 个时钟脉冲保持不变。状态机将在状态 P 和 L 之间不确定地振荡。从问题 10.11 的解：

OUT_1 = L + P（无条件：OUT_1 = 1 如果状态机是在状态 L 或 P）

OUT_2 = Q + LABD + NCD

OUT_3 = R（无条件）

输出波形如图 10-30 所示。

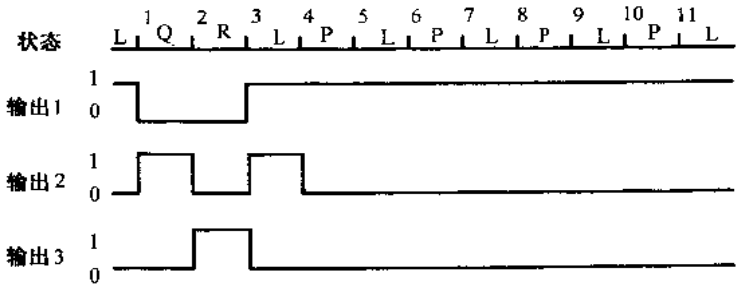


图 10-30

第 4 个脉冲之后即使 L 复发，输出 2 仍保持为 0；因为它是条件性地取决于输入 A。输入 A 在整个范围之外它保持为 0。

10.14 在例 7.5 的系统中（图 7-18）。两个触发器在同一时刻加上时钟，因为信号路程长度的差别，时钟脉冲到达每个触发器的时间稍有不同，它们被设成是时滞（skewed），除了例 7.5 给出的数据外，最小定时参数提供如下：

触发器	门
$T_{pd,HL}$ = 7 ns	$T_{pd,HL}$ = 4 ns
$T_{pd,LH}$ = 5 ns	$T_{pd,LH}$ = 4 ns
T_{su} = 0 ns	
T_h = 0 ns	

求:所允许的最大时钟时滞。

解 考虑最早时钟脉冲到达的时间顺序。它引发一个相关的触发器,该触发器引起逻辑输入的变化并依次引起逻辑输出的变化。这个过程用图表示在图 10-31 上。5 + 4 + 4 = 13ns 之后,D 输入端就可能包含错误数据,这些错误数据会被后面的时钟脉冲锁存。这个可允许的最大时钟时滞就是 13ns。

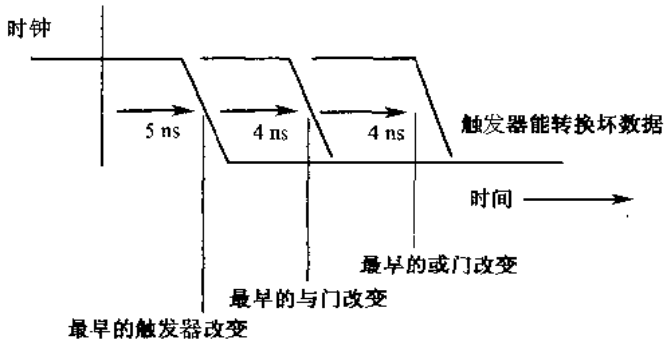


图 10-31

补 充 习 题

10.15 一个带有 ROM 组合逻辑的状态机计数器简化图如图 10-32 所示。希望在每一个标着 X 的行插入一个方框,框内包含的逻辑电路,在 L = 1 时,并行装入数据到所有的 D 触发器。当 L = 0 时,处理计入的数据。设计这个方框内(图 10-33)的逻辑电路。使用尽可能少的门。

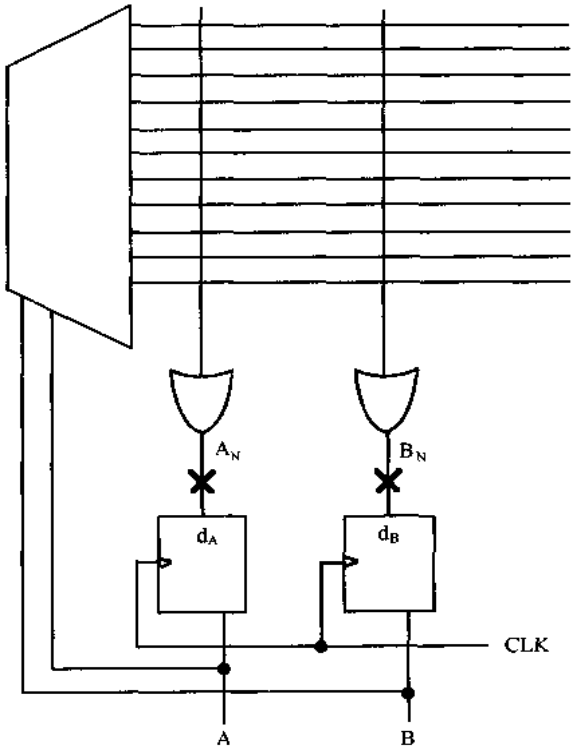


图 10-32

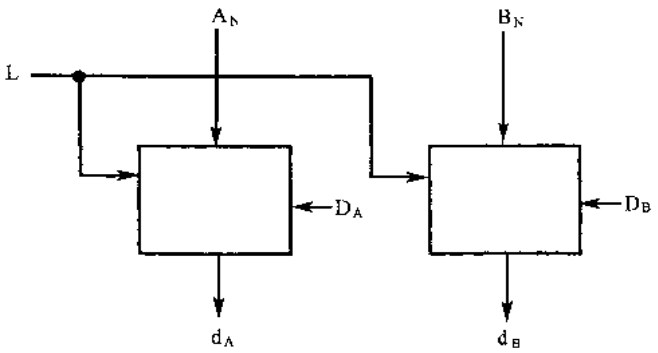


图 10-33

10.16　用 JK 触发器和所需要的门实现表 10.20 计数顺序的状态机计数器，要求使用门的数量最少，并且给出所设计的逻辑方程(参考图 10-19)。

表 10.20

A	B	C
0	0	1
1	0	0
0	1	1
1	1	0
1	1	1

重复

10.17　对表 10.21 给出的时序重复问题 10.16

表 10.21

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

重复

10.18　为了对比的目的，用 SSI 门和 JK 触发器，实现问题 9.17。

10.19　求按表 10.22 顺序计数的状态机的布尔方程。假设 D 触发器用于寄存器的状态。

表 10.22

C	B	A
0	0	0
0	1	0
1	1	0
1	0	1
0	0	1
1	1	1
0	1	1
1	0	0

10.20 下面所示是状态表输入程序的一部分。求相应的表。提示：只是切合程序要求的那部分表被列出来。

```
A:
    IF X + Y THEN B
    A
B:
    C
    ASSERT:
    OUT2
C:
    CASE
        X:A
    ENDCASE
    B
    ASSERT:
        IF X' THEN OUT1
        IF X THEN OUT2
END $
```

10.21 使用图 10-34 所示的 ASM 表，画出一个单状态控制器的状态转换表，写出设计方程。

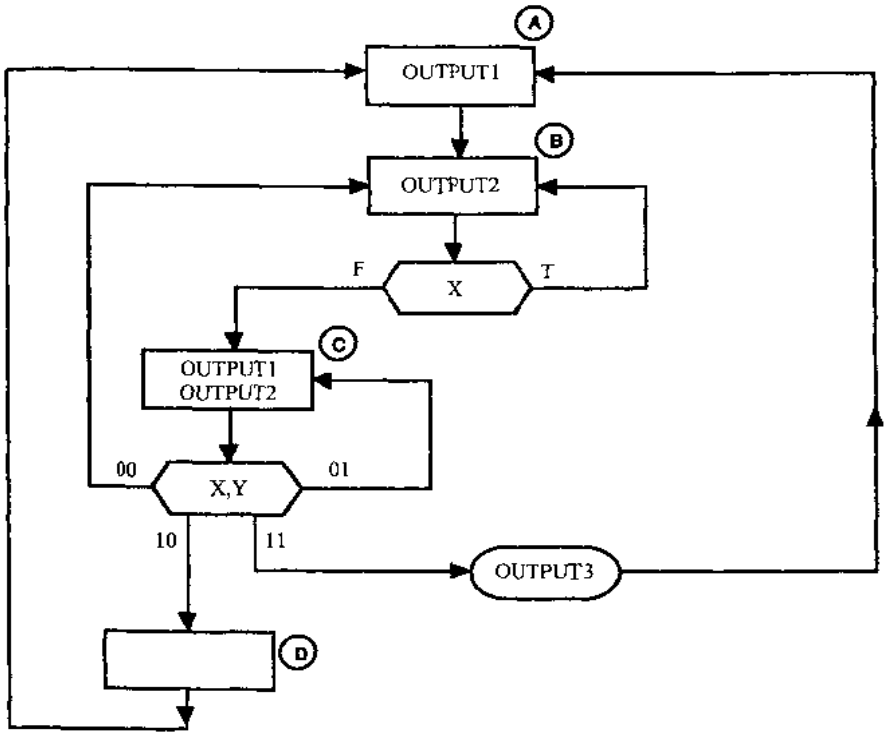


图 10-34

10.22 使用前面问题的 ASM 表，求最小触发器状态机的设计方程。

10.23 重复问题 10.22，使用 ispLSI 软件写出状态表输入程序。

10.24 画出如图 10-35 所示对应的一个单状态控制器的状态转换表。

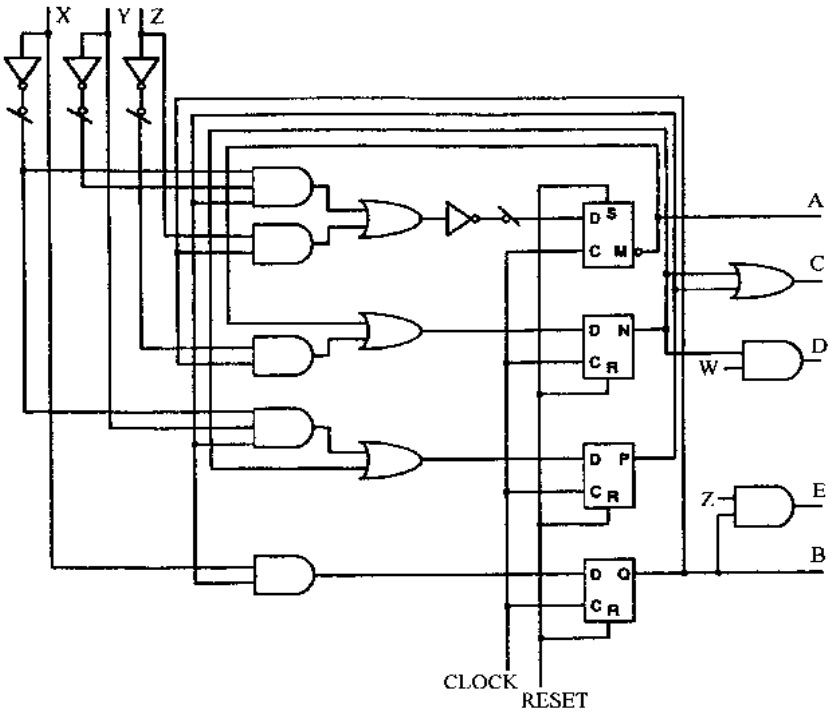


图 10-35

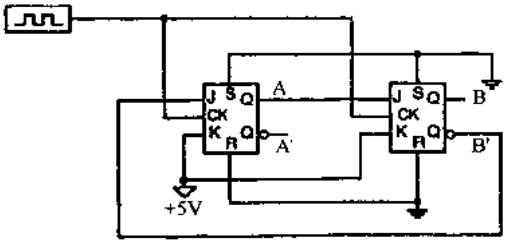


图 10-36

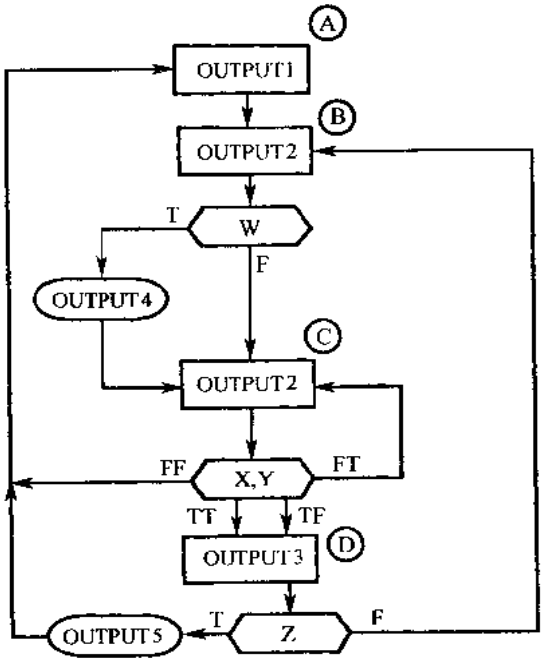


图 10-37

10.25 重新安排问题 10.24 的状态表，成为适合得到最小触发器实现的形式，并对状态写出设计方程。

10.26 一个小状态机如图 10-36 所示，写出相应的 ASM 表，指定 A 和 B 为状态触发器。它将有助于状态机在某一指定状态启动，并检查当时钟脉冲继续进行时的执行情况。

10.27 写出图 10-37 所示的 ASM 表，画出状态转换表，并写出用最少触发器实现的一组状态方程。

10.28 使用问题 10.27 中的 ASM 表，完成时域图，如图 10-38 所示，假设系统初始于状态 A。

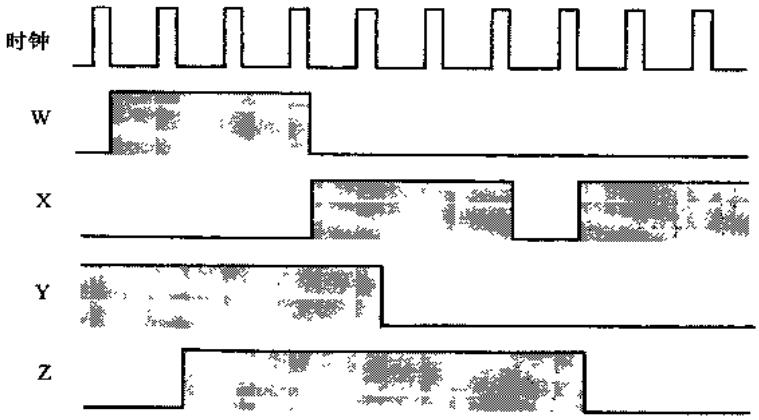


图 10-38

10.29 使用 F 为初始状态，对一个单状态控制器设计的问题 10.8 重复设计。

10.30 使用 JK 触发器代替 D 触发器，重复问题 10.29 的设计。

补充习题答案

10.15 见图 10-39，提示如果 L′不可得到，要求设计者必须考虑硬件反相器能产生一个信号条件。

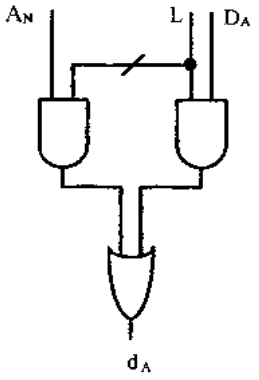


图 10-39

10.16 要求 3 个 JK 触发器，有如下输入：

$$J_A = 1 \qquad K_A = C + B'$$
$$J_B = A \qquad K_B = AC$$
$$J_C = 1 \qquad K_C = A'$$

如果得到触发器的两个输出，只需要 2 个门。

10.17 JK 输入是：

$$J_D = ABC \qquad K_D = A$$
$$J_C = AB \qquad K_C = AB$$
$$J_B = AD' \qquad K_B = A$$
$$J_A = 1 \qquad K_A = 1$$

10.18 一种合理的方法是对每个计数变量 A, B, C 和 D(F=0)要得到一个 J 图和一个 K 图。并对 F=1 重复此过程。从 16 个图得到：

$$J_{AN} = F'B + FB'C'$$
$$K_{AN} = FAB'$$
$$J_{BN} = F'A + F'D + FC' + FA$$
$$K_{BN} = F'C + FA' + F \oplus D$$
$$J_{CN} = F'AB + F'A'D' + FA'B'$$
$$K_{CN} = F'AB + FA'B' + FAD$$
$$J_{DN} = F'A' + F \oplus B$$
$$K_{DN} = FA + F \oplus B$$

10.19 未化简的方程式是：

$$d_A = AB'C' + AB'C + A'BC + ABC$$
$$d_B = A'B'C' + AB'C' + A'BC' + ABC$$

$$d_C = AB'C' + A'BC' - ABC' + A'BC$$

10.20　见图 10-40

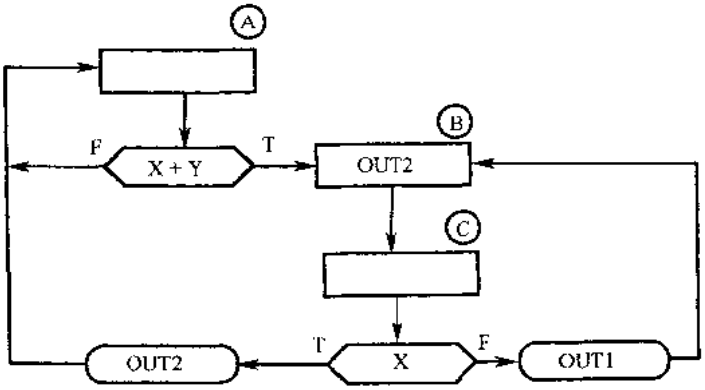


图 10-40

10.21　见表 10.23

表 10.23

前态	现态	条件
C	A	XY
D		无条件
A	B	无条件
B		X
C		X′Y′
B	C	X′
C		X′Y′
C	D	X Y′

设计方程：

$dA = C\,X\,Y + D$　　$OUTPUT1 = A + C$

$dB = A + BX + CX'Y'$　　$OUTPUT2 = B + C$

$dC = BX' + CX'Y$　　$OUTPUT3 = CXY$

$dD = CXY'$

10.22　有 4 个状态，因此 3 个触发器将是最小的。见表 10.24。

表 10.24

现态 M N	次态 M N	条件
A　0 0	B　0 1	无条件的
B　0 1	B　0 1	X
	C　1 0	X′
C　1 0	A　0 0	XY
	B　0 1	X′Y′
	C　1 0	X′Y
	D　1 1	XY′
D　1 1	A　0 0	无条件的

设计方程：

$$D_M = BX' + CX'Y + CXY' = M'NX' + MN'(X'Y + XY')$$

$$D_N = A + BX + C(X'Y' + XY') = M'N' + M'NX + MN'Y'$$

$$OUTPUT1 = A + C = M'N' + MN' = N'$$

$$OUTPUT2 = B + C = M'N + MN'$$

$$OUTPUT3 = CXY = MN'XY$$

10.23 下面的答案只给出相关部分。

```
STATES:    [Q_M  Q_N]
  A        [0   0]
  B        [0   1]
  C        [1   0]
  D        [1   1]
A:
  B
  ASSERT:
      OUTPUT1
B:
  IF X THEN B
  IF X' THEN C
  ASSERT:
      OUTPUT2
C:
  IF X'*Y THEN C
  IF X*Y THEN A
  IF X'*Y' THEN B
  IF X*Y' THEN D
  ASSERT:
      OUTPUT1
      OUTPUT2
      IF X*Y THEN OUTPUT3
D:
  A
```

10.24 见表 10.25

表 10.25

前态	现态	条件
P	M	X'Y'
Q	–	Z
M	N	TRUE
Q		Z'
N	P	TRUE
P		X'Y
P	Q	X

10.25　见表 10.26

表 10.26

M	N	0 1	真
N	P	1 0	真
P	P	1 0	X′Y
	Q	1 1	X
	M	0 0	X′Y′
Q	N	0 1	Z′
	M	0 0	Z

设计方程：

$$d_U = N + P(X + X'Y) = U'V + UV'(X + Y)$$

$$d_V = M + PX + QZ' = U'V' + UV'X + UVZ'$$

10.26　见表 10.27 和图 10-41

表 10.27

现态						次态	
A	B	J_A	K_A	J_B	K_B	A	B
0	0	1	1	0	1	1	0
1	0	1	1	1	1	0	1
0	1	0	1	0	1	0	0
1	1	0	1	1	1	0	0

提示：状态机本身不能假设状态为 11，如果有一个外部信号，就近输入，它只能得到此状态。

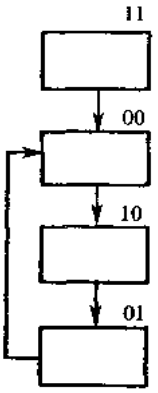


图 10-41

10.27　两个触发器，要求表示成 M 和 N(见表 10.28)

表 10.28

现态(MN)		次态(MN)		条件
A	0 0	B	0 1	真
B	0 1	C	1 0	真
C	1 0	A	0 0	X′Y′
		C	1 0	X′Y
		D	1 1	X
D	1 1	A	0 0	Z
		B	0 1	Z′

10.28　见图 10-42

10.29　控制器方程：

$$D_F = N + M'$$

$$D_G = F$$

$$D_H = GX' + LW$$

$$D_J = GX$$

$$D_K = JV$$

$$D_L = HX$$

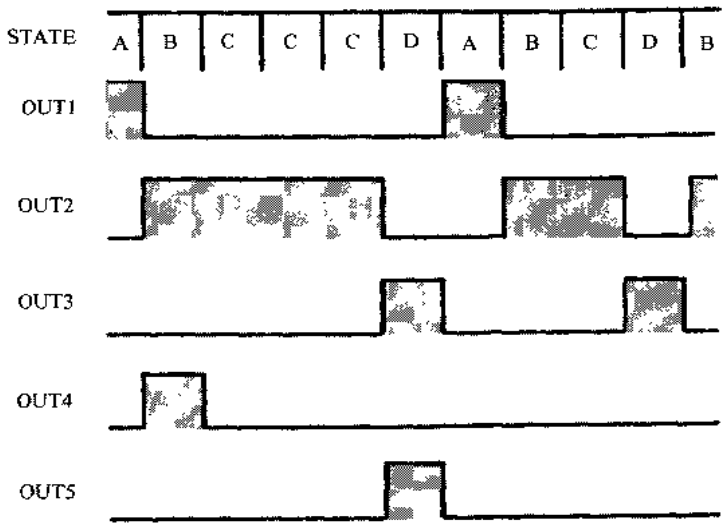


图 10-42

$$D_M = HX'W' + KZ'$$
$$D_N = LW' + HX'W$$

输出方程：

$$A = F + H + L$$
$$B = H + J + K$$
$$C = G + M + HX'$$

10.30 在一个单状态控制器的转换电路中，对于被选中的状态为 0 到 1，当不被选中时为 1 到 0。否则：触发器失效为 0。使 K 总等于 1，当期望跳变从 0 到 1 时，则使 J = 1。对于初始状态 J 和 K 的作用相反。

控制器方程：

$$K_F = N + M \qquad J_F = 1$$
$$J_G = F \qquad K_G, \cdots, K_N = 1$$
$$J_H = GX' + LW$$
$$J_J = GX$$
$$J_K = JV$$
$$J_L = HX$$
$$J_M = HX'W' + KZ'$$
$$J_N = LW' + HX'W$$

输出方程：

$$A = F + H + L$$
$$B = H + J + K$$
$$C = G + M + HX'$$

第 11 章　电可编程功能

11.1　引言

在计算机系统中，常常借助 1 个或多个控制信号改变所使用电路的功能，与第 9 章讨论的可编程逻辑器件相比，电输入信号改变了电路功能却没有引起任何硬件上的物理变化发生。

11.2　基本元件

异或门

异或门常用来作为受控反相器，数据加到它的两个输入端之一，而控制信号加到另一个输入端，如图 11-1 所示。对应的逻辑方程和相关真值表已给出。我们看到，当控制信号是 0 时，数据输出和输入是一致的。而当控制信号是 1 时，数据输出变成数据输入的逻辑反。控制输入因此被看成为决定输入是直接还是以逻辑求反的形式通过该器件。

控制	数据输入	数据输出
0	0	0
0	1	1
1	0	1
1	1	0

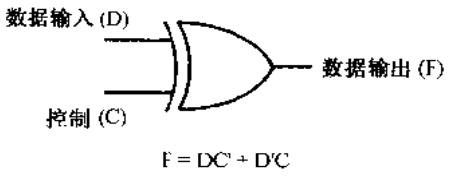


图 11-1　异或门作为受控反相器

注意，当设计的目的是实现一个可编程器件时，虽然这个门的操作被习惯地描述为布尔形式，它更有用的是转变 1 的观念，在目前情况下将其看作为一个可编程的反相器。

二输入逻辑门

基本与门可以看成一个受控开关。象上述那样，一个输入被表示成控制，另一个看作数据端。图 11-2 所示是其结构、逻辑方程和真值表。如果控制输入为 1，则数据直接通过门，如果控制信号是 0，输出是 0，无数据传输。在正逻辑中，2 输入与非门能被看作受控的反相开关。

同样，二输入或门可用作受控开关，在这种情况下，C = 0，输出得到传输数据的信号，而 C = 1，在输出端为 1，与数据输入无关。

控制	数据输入	数据输出
0	0	0
0	1	0
1	0	0
1	1	1

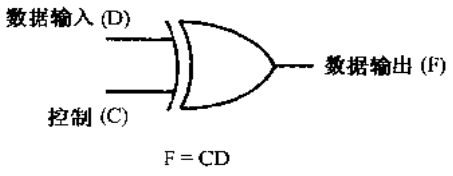


图 11-2　与门视为受控开关

多路选择器

多路选择器作为一个受控开关使用已在第 5 章中讨论过。当相应的控制信号加到它的选择线上时，该器件可用于多个输入中选一。

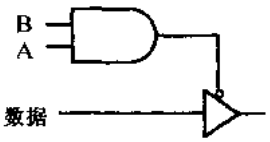


图 11-3　逻辑控制的三态输出器件

三态器件

二态门器件(前面第 9 章中讨论)也被用来作为受控开关。当使能时,它实现一个反相器或者一个缓冲器的功能;当被禁止时,其输出总为开路状态。如果使能输入端的状态由图 11-3所示的某一逻辑门决定的话,附加的灵活性就能得到。在这种情况下,LT 低电平选通的三态门由两个信号(A 和 B)控制;以至只有当 A 和 B 不同时有效时,数据才出现在输出线上。

RAM 查找表

在第 5.4 节,对只读存储器(ROM)作了逻辑上的描述。已学到了这样的应用,即 ROM 基本上用作一个查找表,其真值表输入起到寻址作用并对相应的输出提供对应的存储内容。从用户的角度来看,该器件可实现组合逻辑。也可回顾一下 RAM 存储的内容,象第 8 章讨论的那样,可以改变电子线路的速度。

我们可以通过把 RAM 方框放在一边和把地址位看作逻辑输入以及储存所希望的逻辑功能,对应的输出把这两个功能组合在一起,如果逻辑改变,RAM 能被简单地重新装入数据来改变逻辑。本质上,RAM 能够看成一个实现逻辑的电可编程 EPROM。

11.3　可编程门阵列

如第 9 章中简单的描述,有一种产品介于 PAL 器件、PLA 和全门阵列,叫做现场可编程门阵列(FPGA),有时称为逻辑单元阵列(LCA)。FPGA 与很多结构化器件不同,在电路的相互连接上无任何物理变化要求。

这些器件包含有非常长的移位寄存器(SRAM),在编程期间,可以存储通过编程软件产生的一长串位串。被嵌入的位串在后台循环,并用来控制组成 FPGA 的各种元件之间的连接。通过输入不同的位串,器件可以很容易地重组。

作为 FPGA 的一个例子,讨论 Xilinx 公司提供的 LCA,它由 3 种类型的元件构成:

1. 可配置逻辑模块(CLB)
2. 输入/输出模块(I/OB)
3. 可编程互连点(PIP)

可配置逻辑模块

图 11-4 所示为一个基本功能单元,包括组合逻辑部分,两个 D 触发器和一个共用时钟,以及用来作为选择开关的多路选择器。还有内部控制部分(未画出)通过嵌入的位串操作组成的 RAM 单元。组合逻辑单元,可描述成一个大的矩阵,它能产生有 5 个外部逻辑变量的不同组合和来自两个触发器的反馈信号,CLB 输入能被编程而产生一个或者 2 个包括它的几个输入组的输出变量。

一种选择是允许 5 变量的任何一个都能够产生单函数;另一种选择是允许产生的任何两个函数输入可达 4 个变量(实际上多少变量组合会有一些限制),组合逻辑是利用嵌入的位串作为 RAM 查找表的一部分而实现。这个查找表是从实现的逻辑真值表得到的。器件编程期间位串被下载,执行时是只读过程(像 ROM)。CLB 也包括触发器的直接数据输入端,并且输入端是独立的。

每个触发器的 D 输入端经多路选择器缓冲,直接与数据输入端连接,或者与两个组合逻辑输出中的一个连接。触发器输出侧的多路选择器允许选择为直接输出,或者将来自组合逻辑的输出经寄存后由 CLB 输出。时钟信号能直接或者被反相后加到 CLB 上。外接复位信号,可控的和全局的都接到触发器上。上面讨论过 1 位或 2 位地址型的多路选择器,它们的地

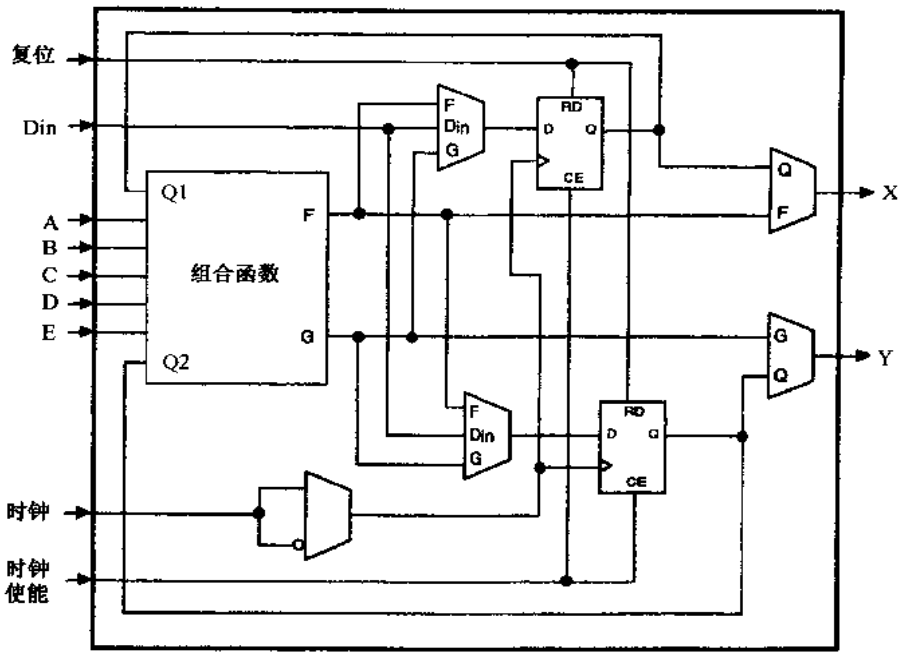


图 11-4　可配置逻辑模块

址输入也全在嵌入位串数据的控制之下。CLB 的输出信号能接到别的 CLB 上作为输入信号，或者被接到 I/O 模块成为芯片的输出。每个 CLB 有直接连到相邻 CLB 的能力，而不用经过开关矩阵。

输入/输出模块

这是一个接到 I/O 端的双向口。在输入位串的控制下，它能配置成一个标准的输出口、三态输出口或者配置成一个输入口。通过多路选择器(作为受控开关)和 XOR(作为受控反相器)的使用，输出信号能直接或者以反相的形式送到这个端口上，也可以寄存方式或者以直接方式输出。当 I/O 模块被配置成一个输入口时，输入信号以直接和以寄存两种形式出现，这些信号或以其中一种方式或以两种方式连接到相应的 CLB 模块。

图 11-5 所示为输入/输出模块的内部逻辑结构，图中顶部的 5 个模块，表示后台 RAM 控制这个指定元件编程的状态。

OUT INV(输出反相)控制触发器，接到一个异或门(G1)上，异或门的其他输入接到 FPGA 输出上。这样，OUT INV 触发器用来作为输出数据的一个控制元件，当它为 1 时，D 触发器 FF1 用反相输出变量下载，当它为 0 时，FF1 以输出变量的原变量方式下载。用同样的方式，TS INV 触发器置为 1 时，允许设计者使用 TS(三态)控制信号的反相形式使能输出缓冲器，因此这个引脚起输出作用。可见，构成的基本元件用于电子控制连接和固定连接。

输出源 RAM 变量决定输出缓冲器是否在被使能时将通过 I/O 模块以直接或寄存型式传输输出变量。

转换速率是指电路响应逻辑电平跳变的速率，以每毫秒(或纳秒)为单位进行电压测量。在目前情况，这个响应时间是在两个预定值之间切换。

无源上拉电阻，当置为 1 时，使输出经过一个电阻连接到电源上，当输出缓冲是开路时(不被使能)，因此“上拉”为正逻辑值。

图 11-5I/O 模块的下部包含的输入信号，被编程连到 FPGA 内的各处。由于输出和输入信号路径共享一个公共引脚，输出能用于内部反馈，到 CLB 或另一个 I/O 模块，输入数据以直接或寄存方式通过。后者经过 FF2 选择 I/O 模块外部的任何一个。

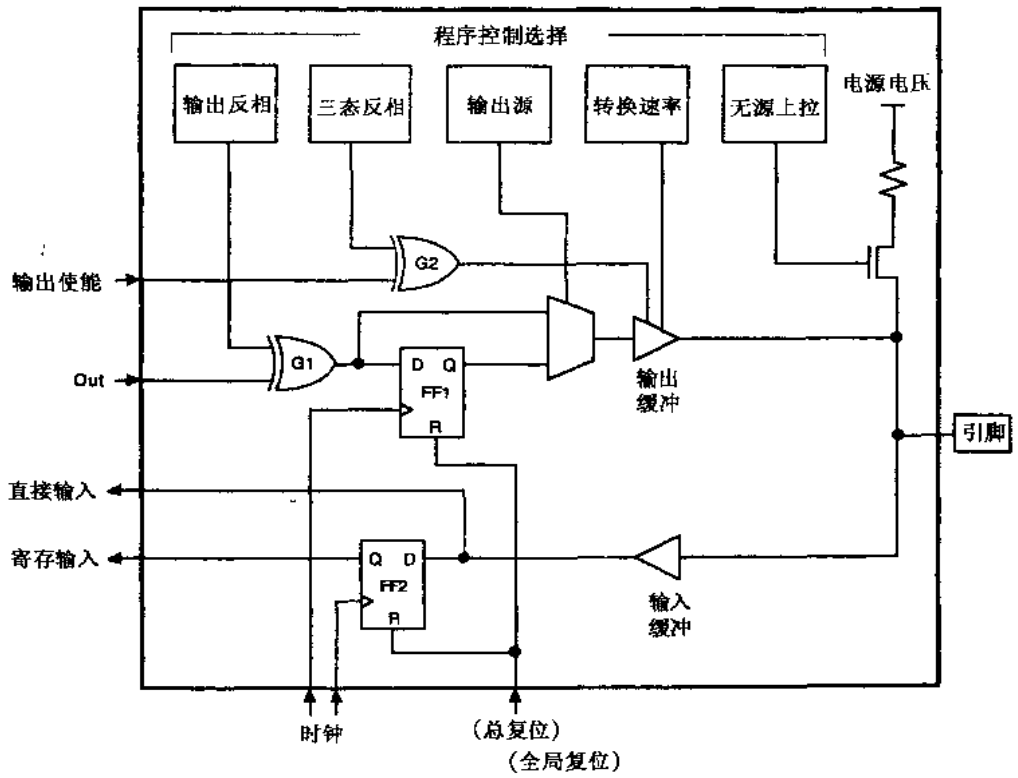


图 11-5　可配置 I/O 模块

可编程互联网络

可编程互联网络(如图 11-6 所示)由多层金属线段网格组成。晶体管开关在嵌入或位串控制下,在所选金属线段和模块端之间形成可编程互联点(PIP)。可以得到 3 种典型的金属资源:

1. 通用互联网络
2. 直接互联

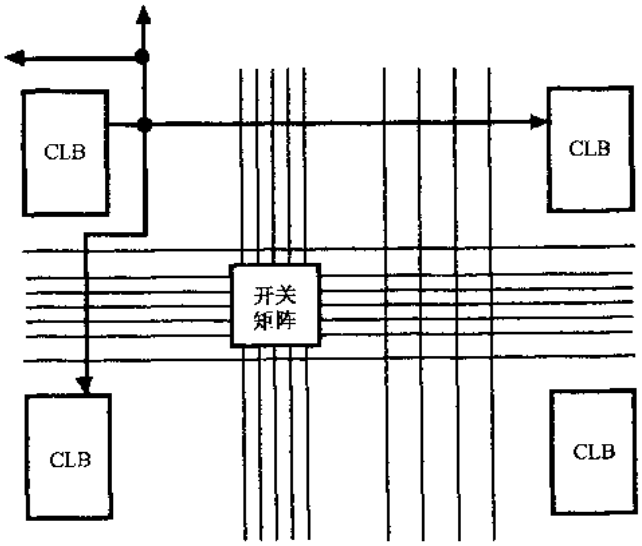


图 11-6　可编程互联网络模块

3．长线

通用互联网是一个开关矩阵，由 5 列垂直和 5 行水平金属线段组成，用 PIP 连接。每条金属线都能够在自身方向上与它正交或平行的线段连接。

直接互联网络为邻近的 CLB 和 I/O 之间的连接提供了路径。

长线旁路开关矩阵在互联的垂直和水平方向上分布。它们使穿越长距离的信号的开关延迟时间最小，并且使信号以最短的路径到达各种目标之间的时滞达到量化的延迟。

程序设计

FPGA 是在计算机控制下编程的。逻辑电路的实现是把逻辑图或以逻辑方程组的形式(同 PAL 器件)输入到计算机中。FPGA 程序要求的附加步骤叫做布局和布线，在布局步骤中，使用一个软件布局算法把逻辑电路和寄存器安排在各个 CLB_s 和 I/O_s 中。然后，计算机进入布线互联步骤，即完成一组可允许的互联路径。在 FPGA 上使用的资源比例很大的情况下，这个过程经常要进行。有时要求熟练人员进行手工操作布线。当使用资源较少时，自动布线和布局就可以满足要求。当程序设计完成时，计算机产生必要的位串，能直接送到硬件器件；而更通常的方法是下载到 PROM 或 EPROM 中，被定位封装到 FPGA 中。

当电源断开时，FPGA 也会把位串丢失，考虑到这种情况，主机系统设计成加电时产生一个信号，FPGA 从 PROM 中自己下载。这种工作方式提供“现场改变”的可能性。FPGA 的配置由 PROM 改变。也可能用一个 FPGA 在工作期间允许使用几个 PROM 重新配置。

11.4　算术逻辑单元

算术逻辑单元通常用在微处理机中，是电可编程器件的另外一个例子。它是使用少量的位数完成多种控制功能的模块化系列组件之一，在本例当中，包含加法、二进制补码减法、加 1、减 1 和逻辑反。这样的模块常称为位片，表示他们可以级连处理较大的位数。ALU(算术逻辑单元)目前可以得到的是内部十分复杂的结构芯片。然而它的工作可通过图 11-7 所示的简化形式很容易理解。

图 11-7(a)所示是一个 ALU 的单片逻辑，图 11-7(b)所示为 4 片相同的 ALU 位片内部互联，应注意的是第 1 片的进位输出作为第 i + 1 片的进位输入。第 0 片的进位输入被接到外部引脚上作为第 3 片进位输出。

控制输入端(MODE，SELECT0 和 SELECT1)在内部分别连接到 3 个门的 1 个输入上，两个输入 A_i 和 B_i 产生一个输出 Fi。输入和输出连接到器件的外部引脚。如果可以得到这样的两个器件，通过连接控制输入在一起，较低位器件的进位输出接到高位的进位输入。2 个 4 位片转换成一个 8 位片。输入和输出之间的关系决定所加的控制信号。方式信号控制片间内部连接，如果它为 0，芯片被隔离，而如果是 1，第 i 个芯片进位输出连接到第 i + 1 片的进位输入。当 MODE = 1 时，电路功能为算术方式，并可权成一个加法器(参考例 11.1)。

SELECT0 被接成一个受控的反相器(XOR)，确定这个芯片是用在 A_i 还是用作逻辑反。SELECT1 连到一个与门上，决定了如果 B_i 被输入到芯片(SELECT = 1)或如果 B_i 被忽略(SELECT1 = 0)，第 0 片的进位输入也可看成是一个特殊的控制输入。如果系统配置成一个加法器(MODE = 1)，到片 0 的进位输入将作为一个初始进位出现，其结果将增加 1。

例 11.1　配置 ALU 实现算术相加。考虑下面的外部连接。

MODE = 1

SELECT0 = 0

SELECT1 = 1

Curry in(0) = 0

下面一步一步地分析，参考图 11-8。

1．MODE = 1，确保 G_1 的输出是 C_i。

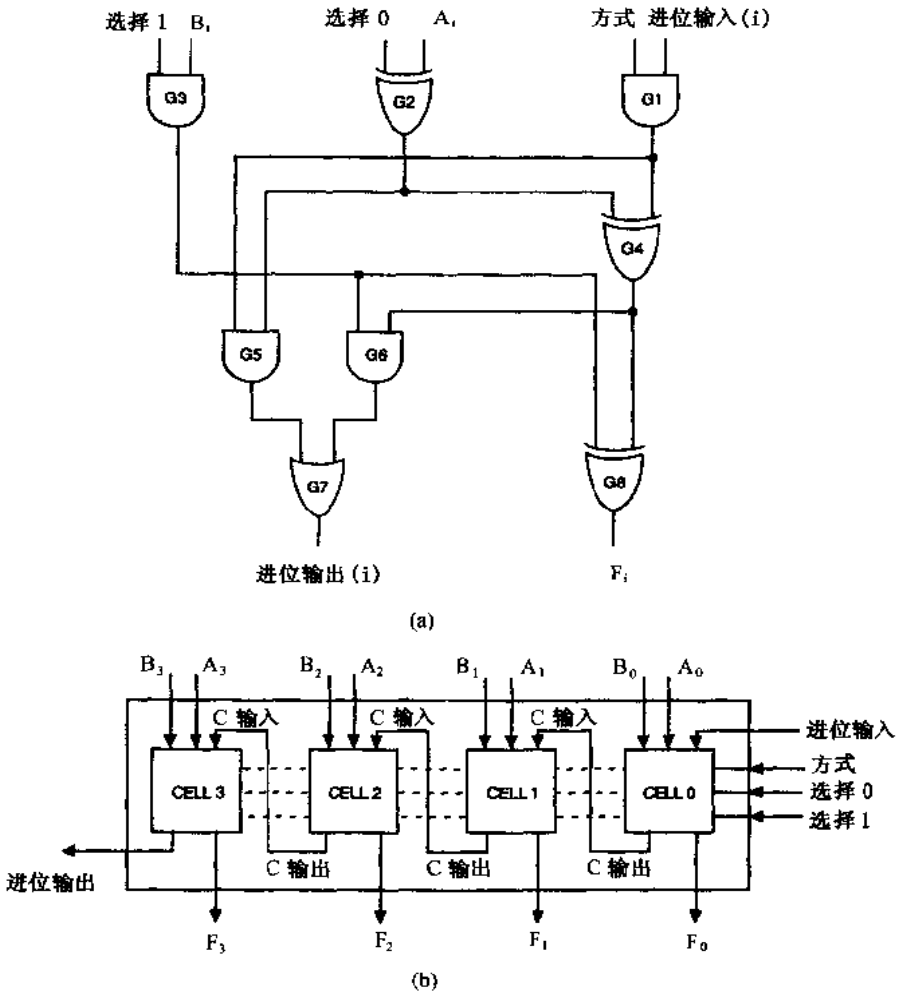


图 11-7　(引自 H. 托博,"数字电路和微处理器"麦格劳-希尔,纽约)

2．SELECT0 = 0,确保 G_2 的输出产生的一个 A_i。

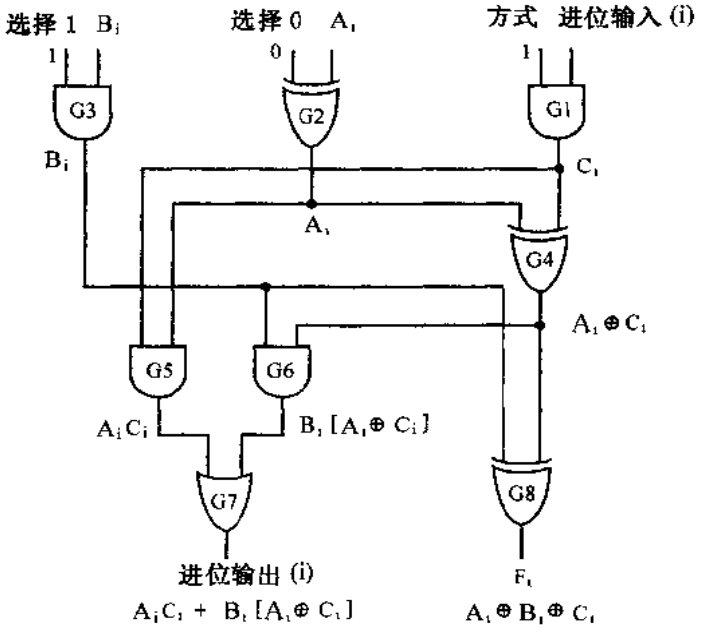


图 11-8　用 ALU 构造加法器

3. SELECT1 = 1，保证 G_3 的输出是 B_i。
4. 输入到 G_4 的是 A_i 和 C_i，因此它的输出是 $A_i \oplus C_i$。
5. 输入到 G_5 的也是 A_i 和 C_i，因此其输出是 A_iC_i。
6. 输入到 G_6 的是 B_i 和 $A_i \oplus C_i$，因此其输出是 $B_i(A_i \oplus C_i)$。
7. G_7 使 G_5 与 G_6 的输出相或，可得到其输出（进位输出）Carry-out(i) = $A_iC_i + B_i(A_i \oplus C_i)$。
8. G_8 的输出是 G_3 和 G_4 异或，因此

 $F_i = A_i \oplus B_i \oplus C_i$

对进位输出和 F 可以创建真值表。

这个真值表（表 11.1）与第 2 章讨论的加法器是相同的。这样系统功能就是一个加法器。

表 11.1

A_i	B_i	C_i	F_i	进位输出(i)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

例 11.2　配置成 ALU 得到二进制数 A 的反码形式。

二进制发的反码形式是对数的每一位求反，将会看到，当 ALU 的控制输入被置成如下状态就可以得到：

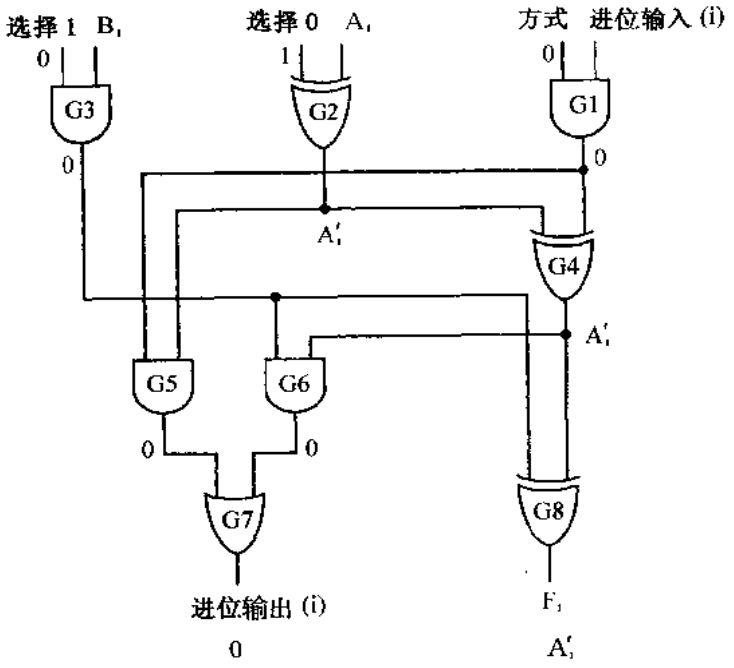


图 11-9　ALU 配置成 1 的反码逻辑电路

MODE = 0

SELECT1 = 0

SELECT0 = 1

与进位输入无关

可用图 11-9 来对电路的性能做一步一步的分析：

1. MODE = 0 保证使 G_1 的输出为 0(隔离此位输入)；
2. SELECT1 = 0 保证使 G_3 的输出为 0(从电路中消掉 B_i)；
3. SELECT0 = 1，使 G_2 输出为 A_i'；
4. A_i' 和 0 输入到 G_4，因此它的输出是 A_i'；
5. A_i' 和 0 输入到 G_5，因此它的输出是 0；
6. A_i' 和 0 输入到 G_6，因此它的输出是 0；
7. G_7 使 G_5 和 G_6 的输出相或，因此它的输出是(进位输出)0；
8. G_8 的输出由 G_3 和 G_4 的输出异或给出，因此 $F_i = A_i' \oplus 0 = A_i'$。

11.5　可编程寄存器

计算机常常利用寄存器在响应各种命令时执行不同的运算。如一个寄存器在程序设计中做与门，如图 11-10 所示。控制和信号线都显示成黑体。寄存器的功能是从总线上取信息，存储起来，并可能对该信息进行操作，然后在适当的程序指令下送回到总线。图中表示出两条总线，是十分常见的同一物理导体。

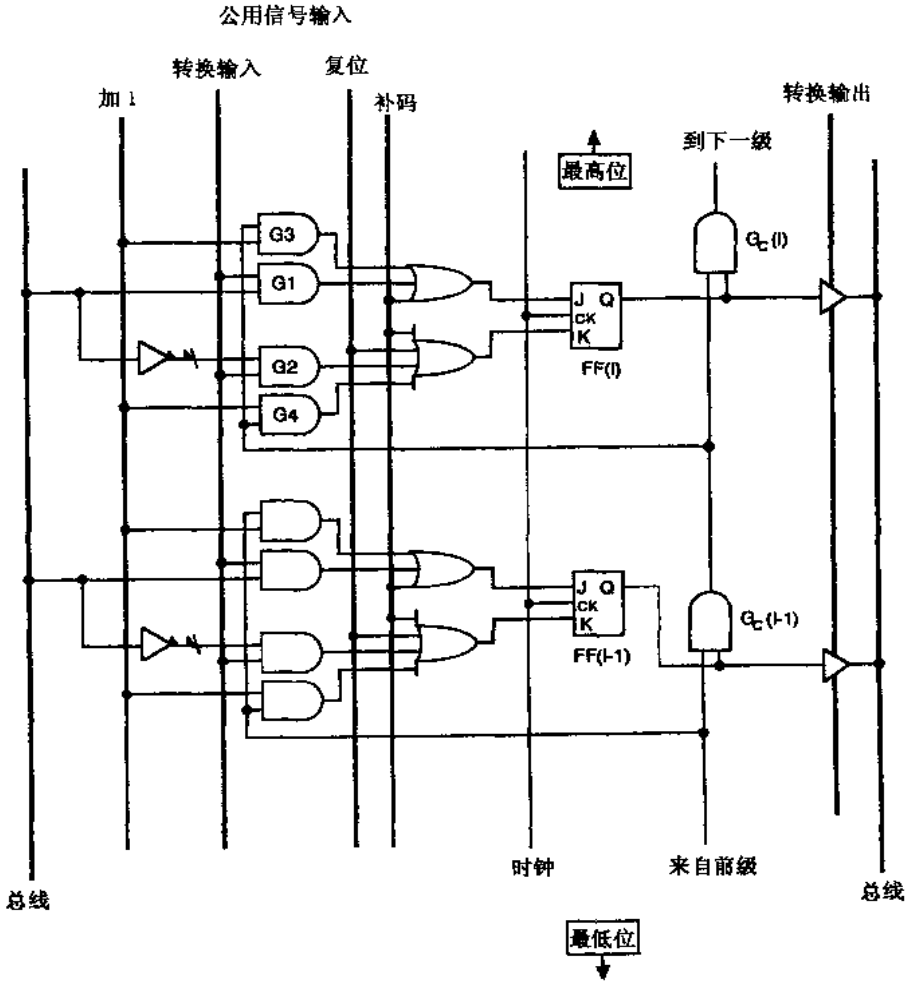


图 11-10

分析寄存器的性能是容易的，只要注意每次只有一个工作，即在任何时刻只有一个命令有效。例如，传输输入信号有效，门 G1 和 G2 的每个状态是重要的。因为加 1 信号是 0，G3 和 G4 在它们的输出端产生 0，这样总线内容被加到 FF(i)的 J 输入端，而它的反被加到 K 输入端，在这种情况下，下一个时钟脉冲产生 Q＝J。同样，如果只是复位线有效，因为是相反的信号所有与门输出全是 0，J 和 K 的输入分别是 0 和 1，下一个时钟脉冲置 Q 为 0，注意转换输出(Transfer out)信号是三态缓冲器的使能信号，缓冲器连接触发器的输出到总线。

在某种程度上对增量运算进行详细讨论是有价值的，当 1 被加到二进制数上，总是最低位翻转。在向高位进位传送过程中，遇到 0 以前，每位都将发生翻转。在这一点上，0 翻转之后此过程停止。遇到时钟脉冲到达，增量过程在所有位同时发生翻转，我们看到给定位翻转，受下一个最低位的值控制，在图 11-10 电路中，这个过程由加 1 控制门[Gc(i)]完成的。如果进行位为 1，将只允许接下去的高位翻转。很明显低位控制门的前级输入应该接到 1 上。

例 11.3　1 0 0 1 1 1 1　加 1。

$$\begin{array}{r} 1\,0\,0\,1\,1\,1\,1 \\ +\qquad\qquad 1 \\ \hline 1\,0\,1\,0\,0\,0\,0 \end{array}$$

其结果与刚才讨论的数相等，从最低位向最高位，包含第一个达到 0 的所有位全翻转。

习 题 解 答

11.1　在图 11-7(a)的 ALU 中，令选择 0 选择 1 方式和进位输入(0)全等于 1，表示出用 4 片芯片实现二进制补码减法运算。

解　这个排列与例 11.1 讨论的二进制加法器电路几乎一样。本例的不同之处是使选择 0 和进位输入(0)为 1。

用 SELECT0＝1 作为输入控制反相器，意味着变量 A_i' 而不是 A_i 将加到 B_i，而且设置 Carry-in(0)＝1，使结果加 1，因此就得：

$$\begin{aligned} F_i &= B_i(+)A_i'(+)1 \\ &= B_i(+)(A_i\text{的二进制补码}) \\ &= B - A \end{aligned}$$

这里符号“(＋)”用来区别逻辑或函数功能和算术加。见题 11.5 关于 A－B 的实现。

11.2　如图 11-10 所示，说明寄存器求补的作用。

解　在任何时间，只有一个控制信号有效，因此

求反(C) ＝ 1
加(I) ＝ 0
传输输入(TI) ＝ 0
复位(R) ＝ 0
传输输出(TO) ＝ 0

关于 J 输入的布尔方程为：

$$J = C + (TI)(BUS) + Gc(i-1)(I) = 1 + 0 \cdot BUS + Gc \cdot 0 = 1$$

$$K = C + R + (TI)(BUS) + Gc(i-1)(I) = 1 + 0 + 0 \cdot BUS' + Gc \cdot 0 = 1$$

因为 J，K 都有效，触发器将翻转，这样使求补值存于寄存器中。

11.3　改变图 11-10 所示的寄存器电路，使电路按命令完成减 1 运算。

解　从一个数中减 1。

0 0 1 1 1 0 0 1 0 0 0

−1

0 0 1 1 1 0 0 0 1 1 1

提示：首先考虑寄存器的内容从右向左迁移，含 1 的位都发生转换。

这个电路可以同样的方法，将当前的累加 1，改变成累加 0。增加两个附加的门，到每级与 JK 输入相关项的与门树。并从附加减 1 控制门[Gd(i)]得到有关它们的输入(如图 11-11 所示)。触发器的 Q′侧得到的信号将确定在已存储的二进制数中 0 的出现。附加电路(用黑线来表示)不影响 其他运算，因为在任何确定的时间只有一个控制信号存在。图 11-11 中，加 1 控制门被删掉使电路更清晰。

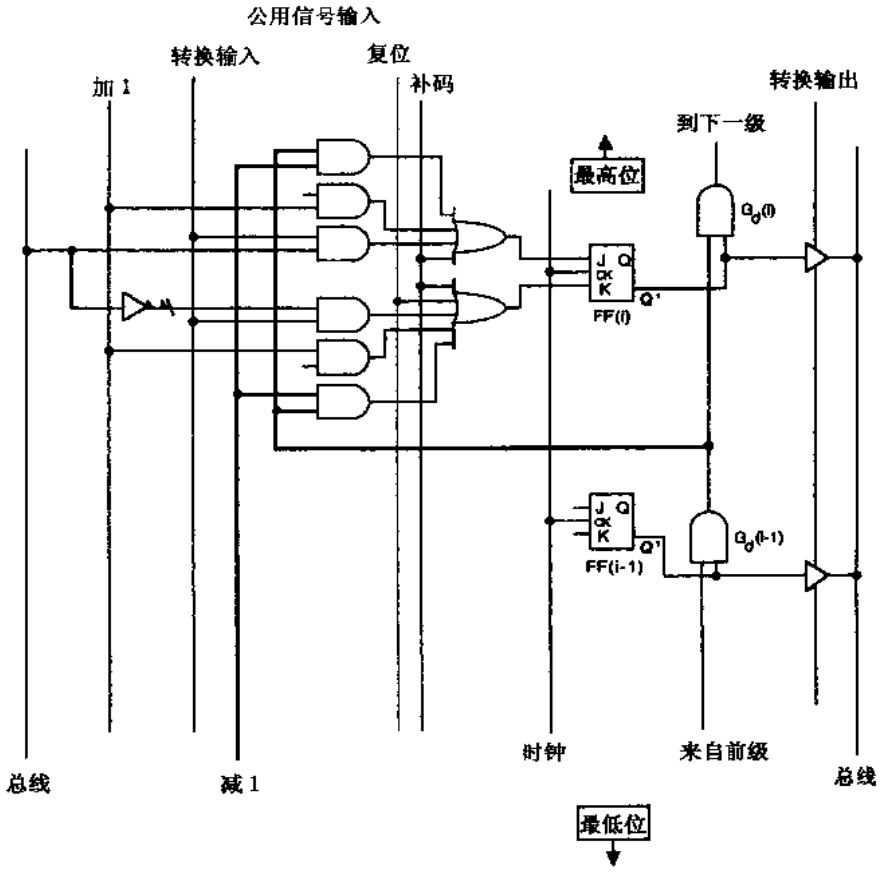


图 11-11

11.4　用图 11-10 所示寄存器，求引起下面事件顺序发生时，控制信号输入的定时关系：

1. 从总线装入数据；
2. 求数据的二进制补码；
3. 将处理的数据传送到输出总线上；

解　提示：二进制数的补码，对每位求反加 1(incrementing)来完成。详细的时序关系如下：

1.0	传送输入信号变为有效。
1.1	时钟脉冲有效，至少 2 级的门延迟之后，加到触发器输入端上并开始建立时间。装入数据。
2.0	时钟脉冲变成无效，传输变成无效，求反变成有效。

续表

2.1　至少一个门延迟使触发器开始建立时间之后，时钟脉冲有效数据求反。
3.0　求反变成无效时钟脉冲变成无效，加 1 变为有效。 3.1　至少两个门延迟使触发器开始建立时间之后，时钟脉冲有效数据加 1。
4.0　加 1 变为无效，时钟脉冲变为无效，传送输出变为有效。数据送到总线上。

11.5　ALU 单元改变的形式，如图 11-12 所示。

SELECT0 = 0
SELECT1 = 1
SELECT2 = 1
MODE = 1
Carry-in(0) = 1

确定所执行的运算及从输入到输出的最大延迟。

解　所给出的电路和问题 11.1 之间的差别仅在多了一个异或门(用 SELECT2 控制)，输入 B_i 通过此门。

如果 SELECT2 = 1，Bi 被求反并且电路产生 Ai(+)Bi′(+)1 等于 A - B。

最大延迟产生在进位输出部分，这里信号通过 4 级门，注意，由于一级的进位输出变为下一级的进位输入。波纹效应发生并且在一个 n 位片的电路中，总的延迟等于 4n 个门的延迟时间。

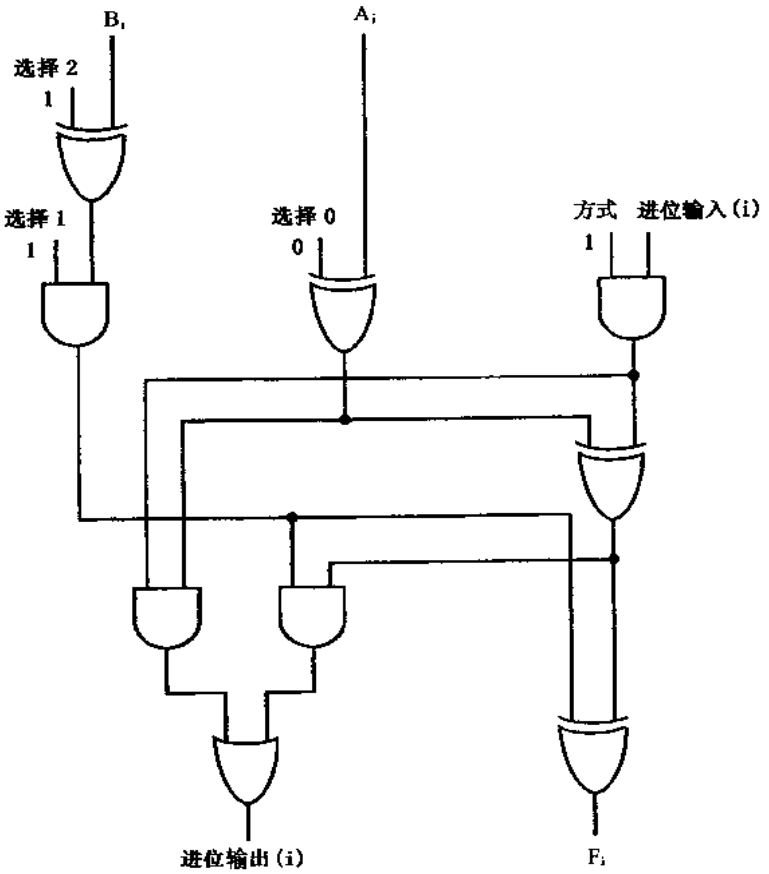


图 11-12

11.6　假设 ALU 的控制信号值设定如下：

SELECT0 = 1
SELECT1 = 1
Carry-in(0) = 1
MODE = 0

说明只有当 A_i 和 B_i 相等时，才有 F = 1 输出。

解 ☞ MODE = 0 意味着没有进位传输，操作按位进行；SELECT0 = 1 意味着 A_i 以反码 A_i' 形式出现；SELECT1 = 1 意味着 B_i 用于直接形式。因为与进位输入(0)无关；与进位输出也无关。

参考图 11-13 所示，$F_i = A_i' \oplus B_i$，构成一个适当的真值表，表示仅当 AB = 00 或 11，F 有效。

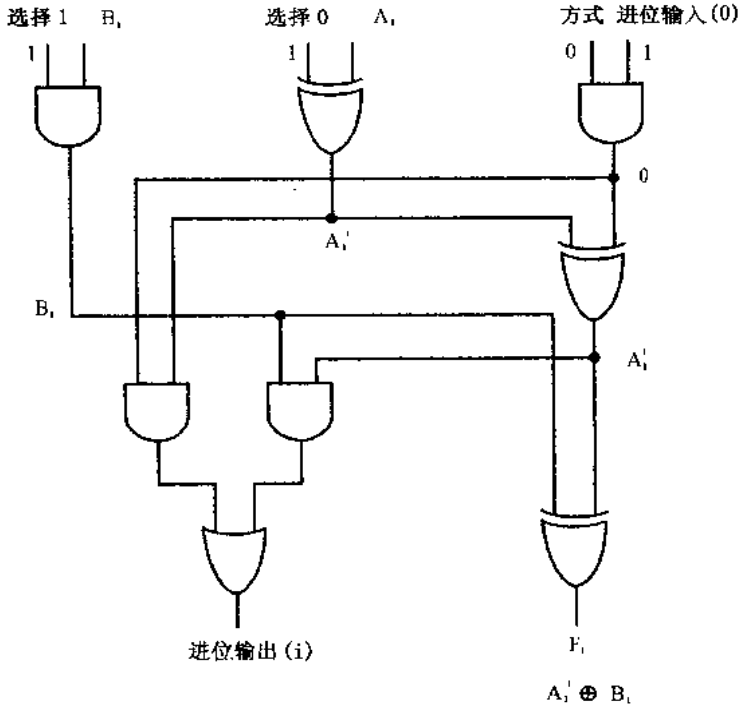


图 11-13

11.7 使用多路选择器设计一个电路，能够用在输出端或者选择信号或者选择它的反码。将结果与第 11.2 节讨论异或门的方法进行比较。

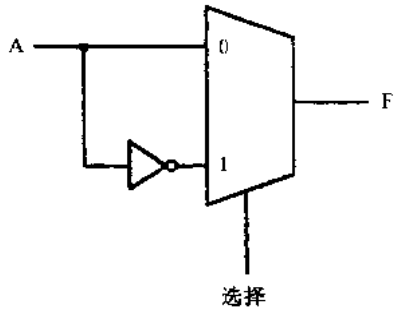


图 11-14

当一个反相器如图 11-14 所示时，二输入多路选择器的选择是从同一变量得到。一个是直接的(A)而另一个则被反相(A′)。

二输入的多路选择器是由两个双输入与门和一个或门组成(见第 5.2 节)。逻辑如图 11-15 所示。

解 ☞ 异或布尔方程由 F = AS′ + A′S 给出，式中 S = select，就得到完全相同的逻辑，象图 11-15 所示的多路选择器的解答一样。多路选择器和异或方法的区别只是在设计者的思想上而不是硅片制造上。

11.8 一个学生设计了一个可编程电路如图 11-16 所示。说明这个设计是可执行的，并且评价其有效性。

解 ☞ S_1 在 A 和 A′之间选择，S_2 在 B 或是 B′之间选择，S_3 决定 A 和 A′是否全部选用。S_4 决定 B 和 B′是否全部选用。

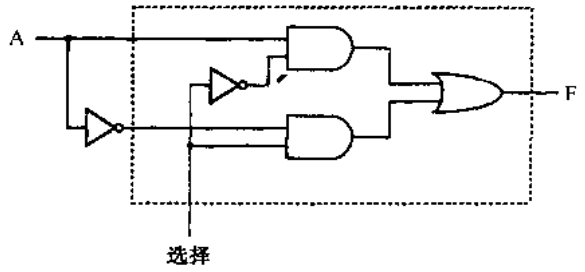


图 11-15

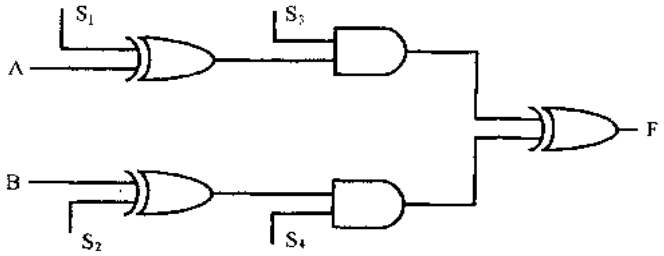


图 11-16

列出所有控制信号构成的输出表(表 11.2)看到有 9 个惟一的输出。

如果它们全是所要用的,那么所有 4 个选择必须使用,因为区分 9 个事件要求有 4 个变量。如果一个输出(比如说,F=0)可以省去,那么应该用 3 个变量而非 4 个就可能实现这个功能。

表 11.2

S_1	S_2	S_3	S_4	F
0	0	0	0	0
0	0	0	1	B
0	0	1	0	A
0	0	1	1	A⊕B
0	1	0	0	0
0	1	0	1	B′
0	1	1	0	A
0	1	1	1	A⊕B′
1	0	0	0	0
1	0	0	1	B
1	0	1	0	A′
1	0	1	1	A′⊕B
1	1	0	0	0
1	1	0	1	B′
1	1	1	0	A′
1	1	1	1	A′⊕B′

11.9　设计一个电路,完成题 11.8 的逻辑功能,0 输出除外。

对策:因为要求 8 个输出,用 3 个选择变量。每种选择组合都要与输出对应。

注意:从表 11.3,S_1 把表分成两组,用这个变量把异或输出和单变量输出区分开。如果

S_2 和 S_3 能够用来产生 A 和 B 及它们的反变量的惟一组合，它们就能被分类以得到单变量并组合成异或逻辑。

表 11.3

S_1	S_2	S_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

解 ☞ 如果 S2 用来作为 XOR 的控制信号，A 作为另一个输入，那么当 $S_2=0$ 时，A 就出现在输出端，当 $S_2=1$ 时，A′就出现在输出端了。同样，S_3 能用来选择 B 和 B′。变量之间的区分需要 S_2 和 S_3 的组合。使用试凑法 S_2 和 S_3 异或(中间输出 Y)。当 Y=1 时，W 被送到 G7(变量 A 选择而变量 B 被禁止)；当 Y=0 时，X 被送到 G7(变量 B 选择而变量 A 被禁止)。见表 11.4。

表 11.4

S_1	S_2	S_3	$W=S_2\oplus A$	$X=S_3\oplus B$	$Y=S_2\oplus S_2$
0	0	0	A	B	0
0	0	1	A	B′	1
0	1	0	A′	B	1
0	1	1	A′	B′	0
1	0	0			
1	0	1			
1	1	0			
1	1	1			

图 11-17 所示电路中虚线框部分，实现了上面讨论的逻辑。门 G_1、G_2 和 G_3 是异或，产生中间输出 W、X 和 Y。$Y=S_2\oplus S_3$ 的非由 G_4 门产生，以控制 A 和 B 变量的选择。变量 B 通过与门 G_5，而 G_6 门通过的变量是 A，G_4 决不允许 A、B 同时通过，因此它们的输出被 G_7 门组合后产生单线输出给最终

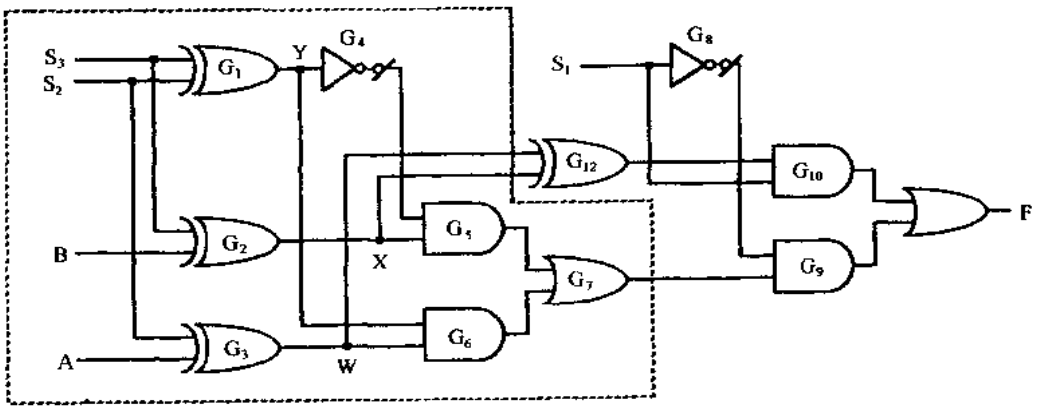


图 11-17

输出逻辑。

注意：元件 G_4、G_5、G_6、G_7 组成的逻辑电路，是两条输入线的选择（是单选择多路选择器），如图 11-15 所示。这个电路结构被重复，用于单变量或者在 A 与 B 异或组合之间进行选择，在此种情况下，当控制变量 $S_1=1$ 时，G_{12}的异或输出通过 G_{10}门到输出 F，而当 $S_1=0$ 时，G_9 允许通过单变量到输出。

提示：虽然这个设计只要求 3 个控制变量，它使用了比问题 11.8 中 4 个控制变量方式大很多的逻辑。这种情况是经常的，减少控制或选择变量数会增加逻辑要求的复杂性。

还应该注意到这个问题解答上的抽象设计方法，理解并使用功能模块，可使其在很大程度上简化，用 5 变量卡诺图方法的逻辑设计，会使逻辑设计相对冗长。一般来讲，从给定的技术要求综合一个电路，可得到多种有效的设计结果，往往没有一种最好的解答。对任何使用和有效的设计方法，设计者的经验和技巧显然是关键因素。

11.10 图 11-18 所示为一个一般性的寄存器结构。假设在任何时刻只有一个输入是有效的（TRUE），写出 J 和 K 的表达式的分析过程，分析下面每种寄存器操作具有何种功能。

1. 信号输入
2. 复位
3. 求反
4. 加 1

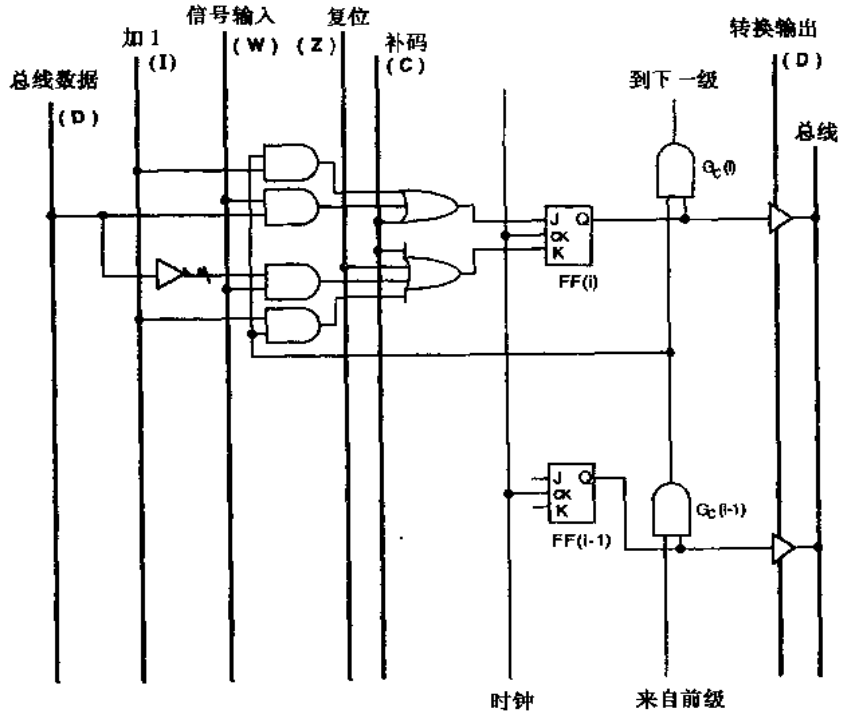


图 11-18

解 一般情况：

$$Ji = C + WD + I[G_C(i-1)]$$

$$Ki = C + Z + D'W + 1[G_C(i-1)]$$

第 1 种情况，信号输入：　$W=1, C=I=Z=0$

$Ji=D$　$Ki=D'$

时钟脉冲之后，$Qi=D$

第 2 种情况，复位（清零）：　$Z=1, W=C=I=0$

$Ji=0$　$Ki=1$

时钟脉冲之后，$Qi=0$

第 3 种情况，求反：　　C = 1　Z − W = I = 0

Ji = 1　Ki = 1

时钟脉冲之后，Qi = Qi′（翻转）

第 4 种情况，加 1：　　I = 1　C = W = Z = 0

Ji = Gc(i − 1)　Ki = Gc(i − 1)

如果 Gc(i − 1) = 1，时钟将引起 Qi 翻转。如果寄存器所有的低位全是 1，将都是这种情况。

11.11　图 11-19 所示是存储器在可编程 RAM 芯片中的信息控制的逻辑电路。芯片 1 控制 S_1，芯片 2 控制 S_2，等等。假设在此之前跟随系统时钟脉冲的数据输入是稳定的，求下面电路的输出和反馈信号：芯片 1，2，3，4 = 0010。

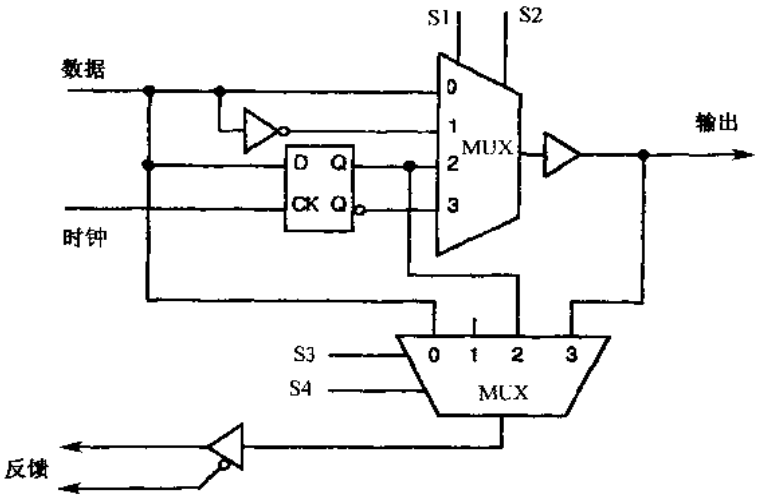


图 11-19　图 9-5 所示放大器型的反馈输出

解　因为芯片 1 和芯片 2 全为 0，$S_1 = S_2 = 0$，且输出跟随数据输入而不受时钟影响。

因为芯片 3 为 1，而芯片 4 为 0，多路选择器线 2 被选择，反馈等于数据的原值或反值，此数据就是最新的时钟锁存到 D 触发器的值。

11.12　考虑问题 11.11 中的逻辑电路。反馈输出的双放大器用异或门来代替，输入之一是反馈多路选择器的输出，另一个是连接到附加 RAM 芯片(5)。如果输出是被寄存输入数据的逻辑非，而反馈信号是未被寄存的数据的逻辑非，求这两种情况下的芯片内容。假设在系统时钟前后数据输入是稳定的。

解　输出多路选择器的线 3 接受了被寄存输入数据的反。这样 S1 和 S2(或芯片 1 和 2)必定两者全是 1。未被寄存数据出现在反馈多路选择器的输入端为 0，所以 S3 和 S4(芯片 3 和 4)一定可编程为 0。如果异或控制输入(芯片 5)被置成 1，就能得到所要求的取反。

11.13　图 11-20 所示输入/输出宏单元，受隐藏的 RAM 控制。为了使信号 4 能够等于来自引脚的输入信号，求 RAM 芯片单元的七位数内容必须是什么？

解　如果芯片单元 3 是 1，MUX2 就禁止三态门缓冲器驱动引脚，使其呈开路状态。如果芯片单元 5 是 0，引脚将连接到信号 4，这样芯片单元的内容(芯片单元 1……7)应该是 XX1X0XX，这里 Xs 表示无关项。

11.14　在问题 11.13 中，如果芯片单元 1，…，7 = 1100100，求其功能。

解　芯片 1 = 1　触发器的 Q 输出被耦合到异或门 G_1 的一个输入上；

芯片 2 = 1　Q 被反相成为 Q′；

芯片 3，4 = 00　三态缓冲器被使能，将 Q′耦合到引脚；

芯片 5 = 1　信号 1 被反馈到信号 4；

芯片 6，7 = 00　信号 2 和信号 3 的值直接决定 Q 的状态。

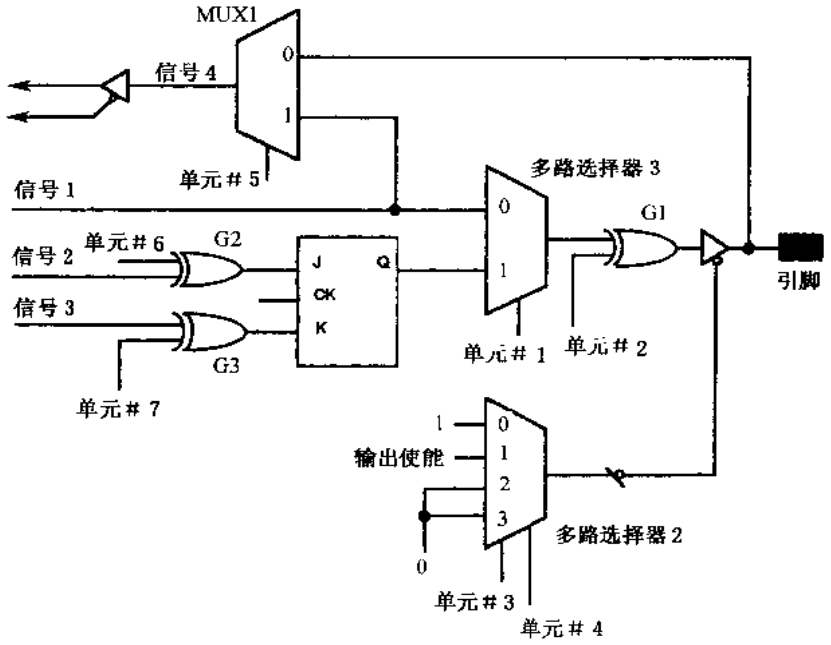

MUX1
信号 4
0
1
单元 # 5
信号 1
多路选择器 3
0
1
G1
信号 2
单元 # 6
G2
J
Q
CK
K
引脚
信号 3
G3
单元 # 1
单元 # 2
单元 # 7
1
0
1
2
3
输出使能
多路选择器 2
0
单元 # 3
单元 # 4

图 11-20

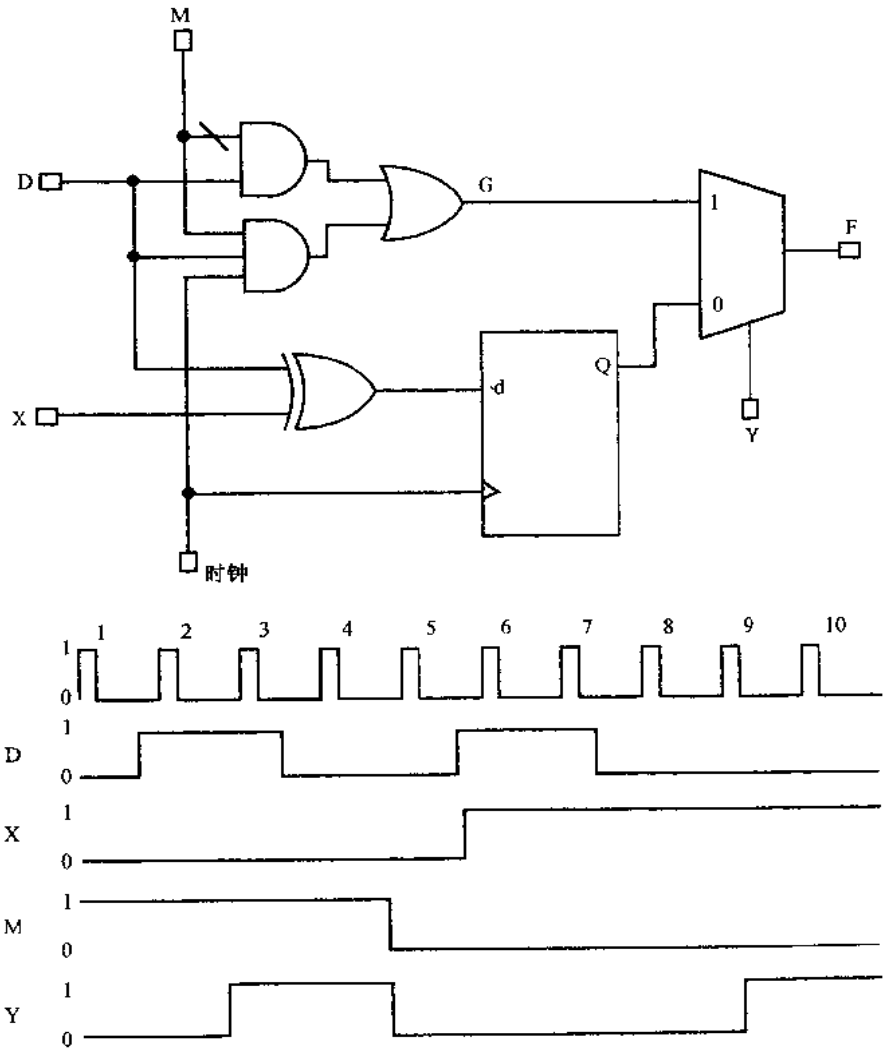

M
D
G
1
F
0
Q
d
X
Y
时钟
1
2
3
4
5
6
7
8
9
10
1
0
D
X
M
Y

图 11-21

11.15　图 11-21 所示为一个可编程电路及其输入波形。说明其工作和画出输出 F 的波形图。

解　当 X = 0 时 Q 是输入 D 的寄存值，而当 X = 1 时，被寄存的是 D′值，当 M = 0 时，OR 门输出 G 等于 D，当 M = 1 时，OR 门输出 G 等于和时钟相与的 D。当 Y = 1 时，F = G，当 Y = 0 时，F = Q。为了画出 F 端的波形图，也给出相关的中间波形 G 和 Q，图 11-22 会有一些帮助。

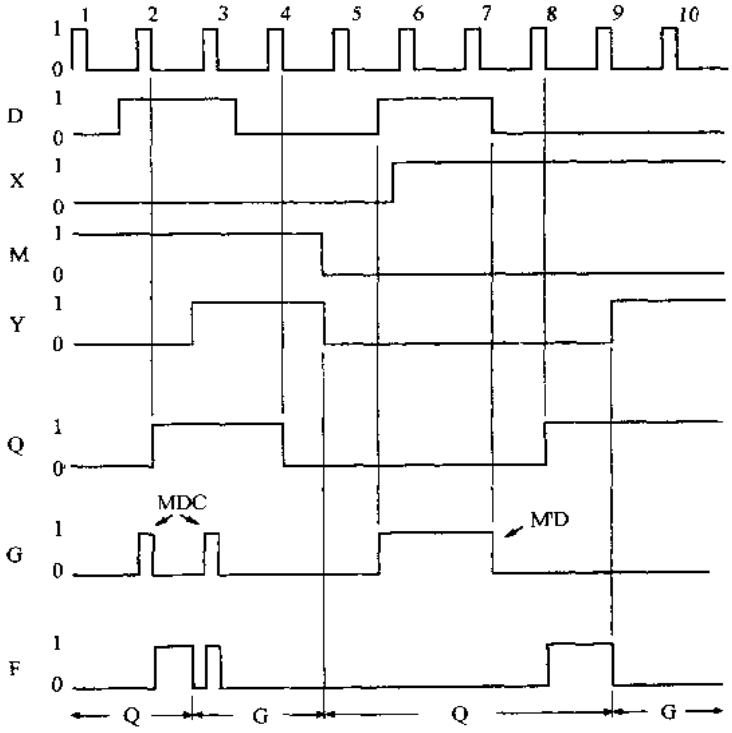


图 11-22

补 充 习 题

11.16　图 11-5 所示 FPGA I/O 模块的 RAM 中存储着如下值：

OUT INV = 1　　output source = 1

TS INV = 0　　passive pull-up = 1

如果 TS 信号输入是 1，并假设多路选择器输入接到选择线是 0。

(a)TS 逻辑值使输出等于什么？

(b)如果输出缓冲器不选通，在引脚端出现的正逻辑为何值？

(c)输出是直接的还是寄存的？

(d)输出是原变量还是反相的？

11.17　对图 11-23 所示的电路，写出输出表达式。如果控制信号被设置成 $S_1 = 1$，$S_2 = 0$ 和 $S_3 = 1$

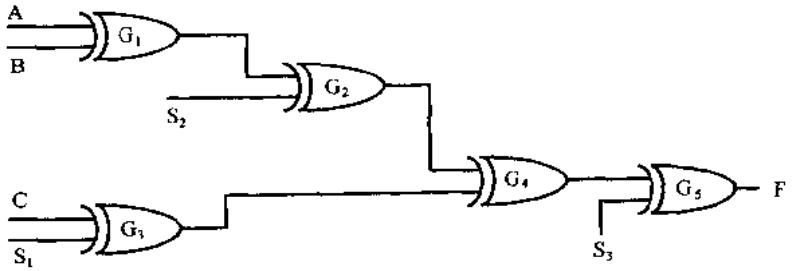


图 11-23

11.18　在图 11-24 所示的电路中，控制字 A 是 $A_1, A_2, A_3, A_4, A_5, A_6, A_7$。字 A = 0111110 的作用是什么？

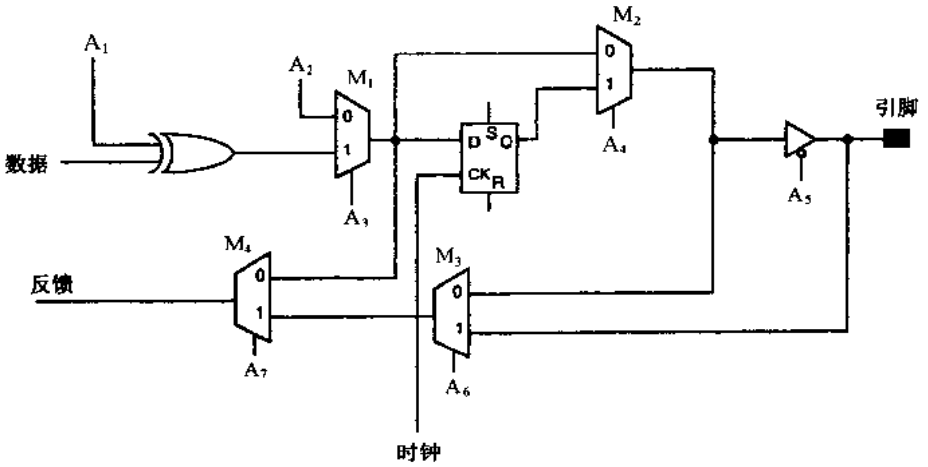


图 11-24

11.19　使用题 11.18 的逻辑图。确定将提供寄存的控制字，在引脚作为输出时的反相数据和反馈的未寄存数据。

11.20　为了测试，常常希望有一个已知值(如图 11-24 所示)在时钟脉冲到来之后放在输出引脚，并反馈同样的值。使用题 11.18 的逻辑来确定合适的控制字。

11.21　题 11.18 的逻辑怎样被改变才可能反馈反相的引脚值？

11.22　图 11-25 的 ALU 控制信号值被设定如下：$S_1 = 1, S_0 = 0, M = 1$，而进位输入为 1，电路具有什么样的功能？

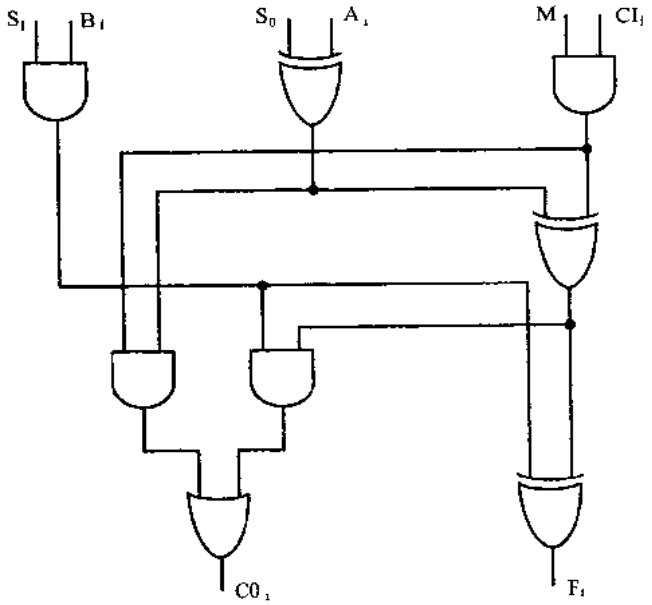


图 11-25

11.23　一个学生，试图得到一个 ALU，少接了几条线，成为图 11-26 所示电路。给定 $M = S_0 = 1$ 和 Carry-in = $S_1 = 0$，求 F 和 C_0 的表达式。

11.24　设计一个电路，使得存在于输入引脚上的信号即一个控制字可以以寄存或不寄存方式以及反相或同相方式通过。

11.25　参考问题 11.24 的解答，创建真值表写出控制字变量所有可能的取值。

11.26　用三态门器件改变题 11.24 的解答，以便使引脚能用于输入或输出。

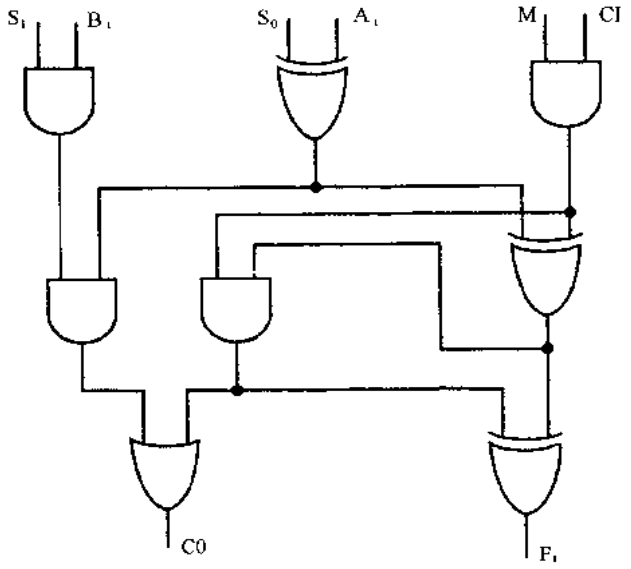


图 11-26

11.27　使用与门改变图 11-27 的电路，以使反馈路径上的触发器能够同步清零。

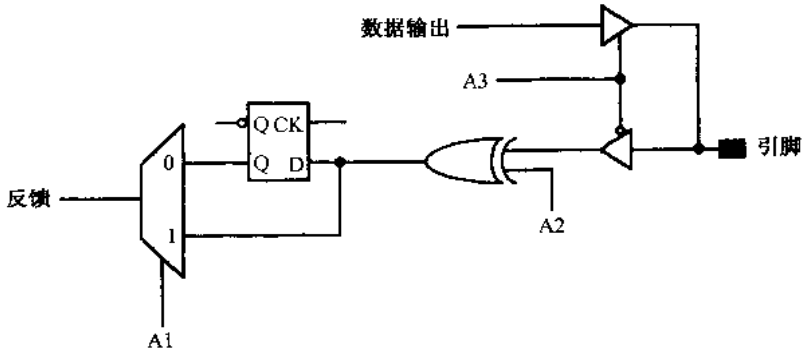


图 11-27

补充习题答案

11.16　(a)使能为 HT，且 TS=1 时将允许输出。

(b)如果输出(不允许)禁止，象是开路。无源上拉是 1，使高电平出现，表示有效。

(c)多路选择器线 1 被选择，并且输出被寄存。

(d)反相

RAM 命令产生一个寄存的和选通的输出并且该输出被反相。

11.17　$F=[(A\oplus B)\oplus C']'$

11.18　$A_1=0$　数据不反相；

$A_2=1$　因为 A3=1，所以无关；

$A_4=1$　第 2 个多路选择器选择被寄存的数据；

$A_5=1$　禁止引脚输出，引脚被设定为输入；

$A_6=1$　选择数据从引脚输入；

$A_7=0$　选择第 1 个多路选择器输出用于反馈并忽略引脚输入。总结：引脚断开，同相输入数据被直接反馈(未寄存)。

11.19　A = 1X110X0　X　代表无关或"无关最小项"位。

11.20　A = X(A_2)01011

11.21　控制变量 A_8 加到受控反相器(XOR),放在 M_3 或 M_4 中任一个多路选择器上的输出。

11.22　A_i(+)B_i(+)1,符号(+)表示算术加。

11.23　$C_o = 0$ 而 $F = A'_i$

11.24　本题有多种可能的解,图 11-28 表示了其中一种。

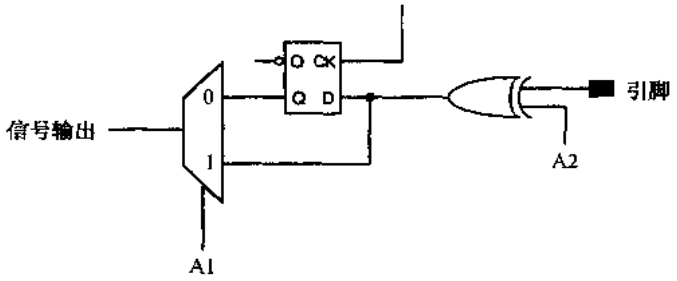


图 11-28

11.25　见表 11.5

表 11.5

A1	A2	反相数据	寄存数据
0	0	不是	是
0	1	是	是
1	0	不是	不是
1	1	是	不是

11.26　参考图 11-27,在这个解答中,一个三态门被置为高电平而另一个为低电平,以至于公用信号 A_3 总可将其中一个或另一个使能。

11.27　见图 11-29,当 A_4 是 0 时,触发器将在下一个时钟脉冲被清零。

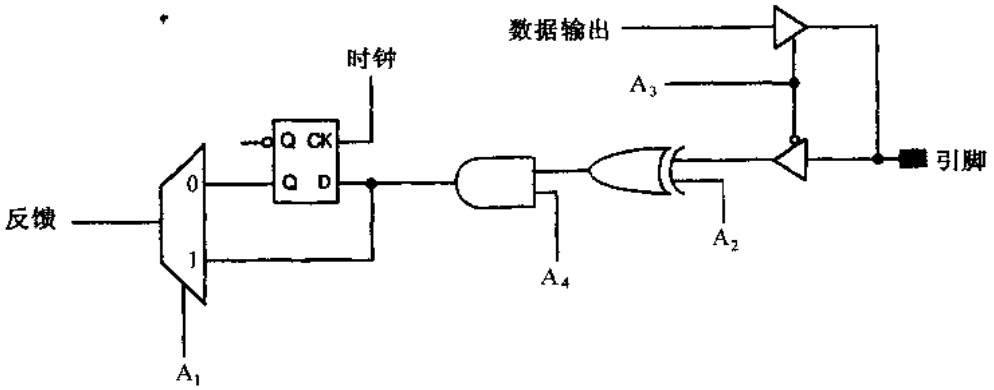


图 11-29

附录 A　基本布尔定理和恒等式

$$A \cdot 1 = A \tag{A.1}$$
$$A \cdot 0 = 0 \tag{A.2}$$
$$A \cdot A = A \tag{A.3}$$
$$A \cdot A' = 0 \tag{A.4}$$
$$(AB)' = A' + B' \quad \text{（狄·摩根定理）} \tag{A.5}$$

在布尔代数中，对偶定理应用意味着在上面一组方程中，如果将每个与改为或，或改为与，1 和 0 互换，就得到第二组恒等式：

$$A + 0 = A \tag{A.6}$$
$$A + 1 = 1 \tag{A.7}$$
$$A + A = A \tag{A.8}$$
$$A + A' = 1 \tag{A.9}$$
$$(A + B)' = A'B' \quad \text{（狄·摩根定理）} \tag{A.10}$$

其他布尔恒等式：

$$(A')' = A \tag{A.11}$$
$$A + AB = A \tag{A.12}$$
$$A + A'B = A + B \tag{A.13}$$
$$AB + A'B = B \tag{A.14}$$

与方程式(A11)到(A14)对应的恒等式：

$$A(A + B) = A \tag{A.15}$$
$$A(A' + B) = AB \tag{A.16}$$
$$(A + B)(A' + B) = B \tag{A.17}$$

3 变量定理：

$$A + BC = (A + B)(A + C) \tag{A.18}$$
$$AB + A'C = (A + C)(A' + B) \tag{A.19}$$
$$(A + B + C)' = A'B'C' \quad \text{（狄·摩根定理）} \tag{A.20}$$

方程式(A.18)到(A.20)对应的恒等式：

$$A(B + C) = AB + AC \quad \text{（分配律）} \tag{A.21}$$
$$(A + B)(A' + C) = AC + A'B \tag{A.22}$$
$$(ABC)' = A' + B' + C' \quad \text{（狄·摩根定理）} \tag{A.23}$$

附录 B　标准逻辑符号

B.1　概述

在写这本书时，设计者和商贩们使用的逻辑符号还没有完全标准化，因此，后来出现某些混乱。这个附录试图说明出现在文献中的各种典型符号并指出某些一般性质。除很多数字逻辑硬件商们的出版物中包括的正逻辑符号之外，还有最新修订的 IEEE/IEC 符号。例如，高级 CMOS 逻辑数据手册，1990 年由德克萨斯仪器公司出版，就给出另一种粗序列的数字集成电路逻辑符号。

B.2　独立的硬件门组

与非逻辑(例，74×00 四-2 输入正与非门)

习惯的引脚-输出符号表示在图 B-1。门的符号表示在封装的轮廓内，表示硬件类型并且

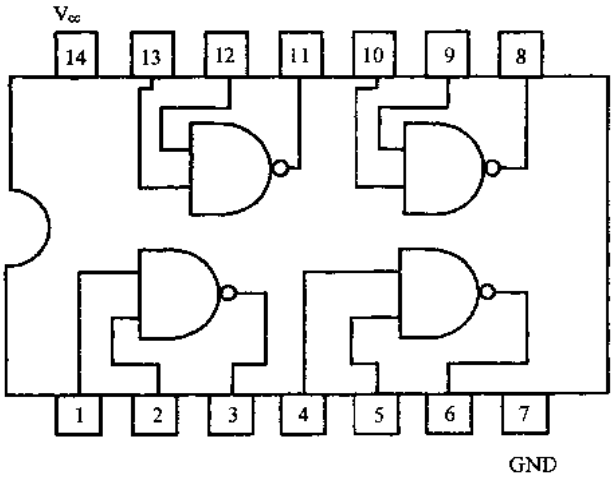


图 B-1　四-2 输入与非门常用符号

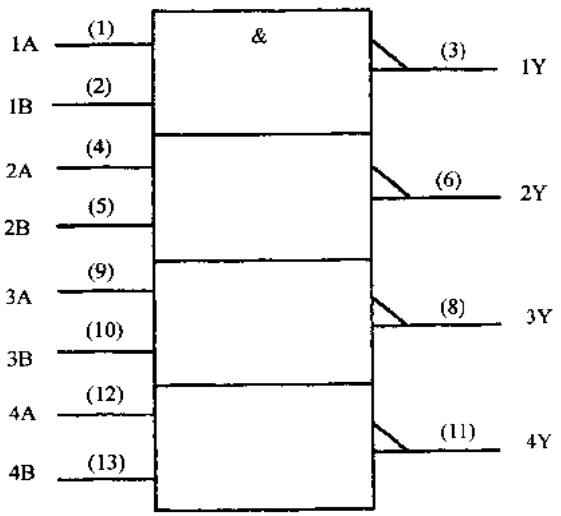


图 B-2　IEEE/IEC 符号表示四-2 输入与非门

一般是"通用的"正逻辑各种类型。从俯视图上看封装的小方块中的数表示引脚号，Vcc 表示电源电压，一般为 +5 伏，而 GND(地)表示电源公共端。与非门硬件混合逻辑符号在第 4 章中讨论过的。

同样的集成电路用 IEEE/IEC 符号习惯表示成图 B-2。这种类型的符号出自国际委员会的慎重考虑，并提出把它变成工业标准。现在说明符号"&"表示与逻辑而括弧中的数字表示封装引脚号，4 个矩形部分(A、B 字母组前标有 1，2，3，4)表示在整个封装中有 4 个相同的电路，每个电路有两个输入标着 A 和 B，1 个输出标 Y。半箭头表示低有效输出。电源引脚在图上未标出。

或非逻辑(例，74×02 四-2 输入正或非门)

习惯的引脚输出图表示在图 B-3。内部的符号表示或非门，象前面的例子那样，从俯视图上看封装的小方块中的数表示引脚号。

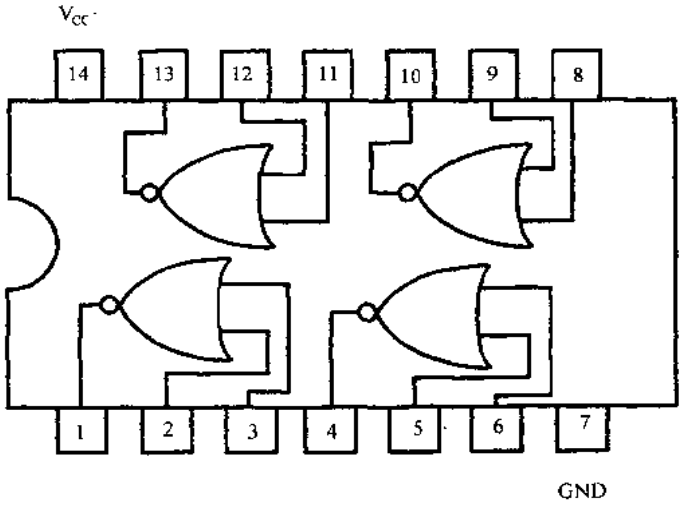


图 B-3　四-2 输入或非门常用符号

图 B-4 所示为四-2 输入或非 IEEE/IEC 逻辑符号。"≥1"表示逻辑或因为 4 个相同电路中的任何一个，每次输入之和为 1，等于或大于 1。半箭头表示低有效输出。电源引脚图上未标出。

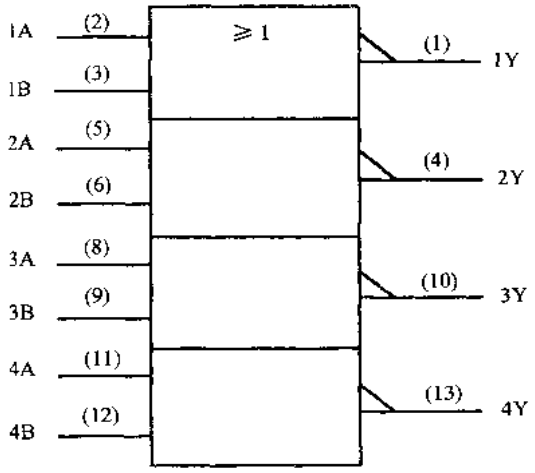


图 B-4　IEEE/IEC 四-2 输入 NOR 逻辑符号

B.3　MSI 逻辑

多路选择器(例,74×153 双 4 输入选 1 数据多路选择器)

这个元件中有两个相同的多路选择器,它们有公用的选择输入端(A 和 B),并且单独被有源低电平信号(1G′和 2G′)选通。输出一般命名为 Y,而数据输入用字母 C 表示。该通用元件能够用逻辑图的习惯符号表示,如图 B-5(a)所示,虽然有些作者更喜欢用图 B-5(b)的三角形方式表示,在很多情况下,用半箭头代替圆圈。

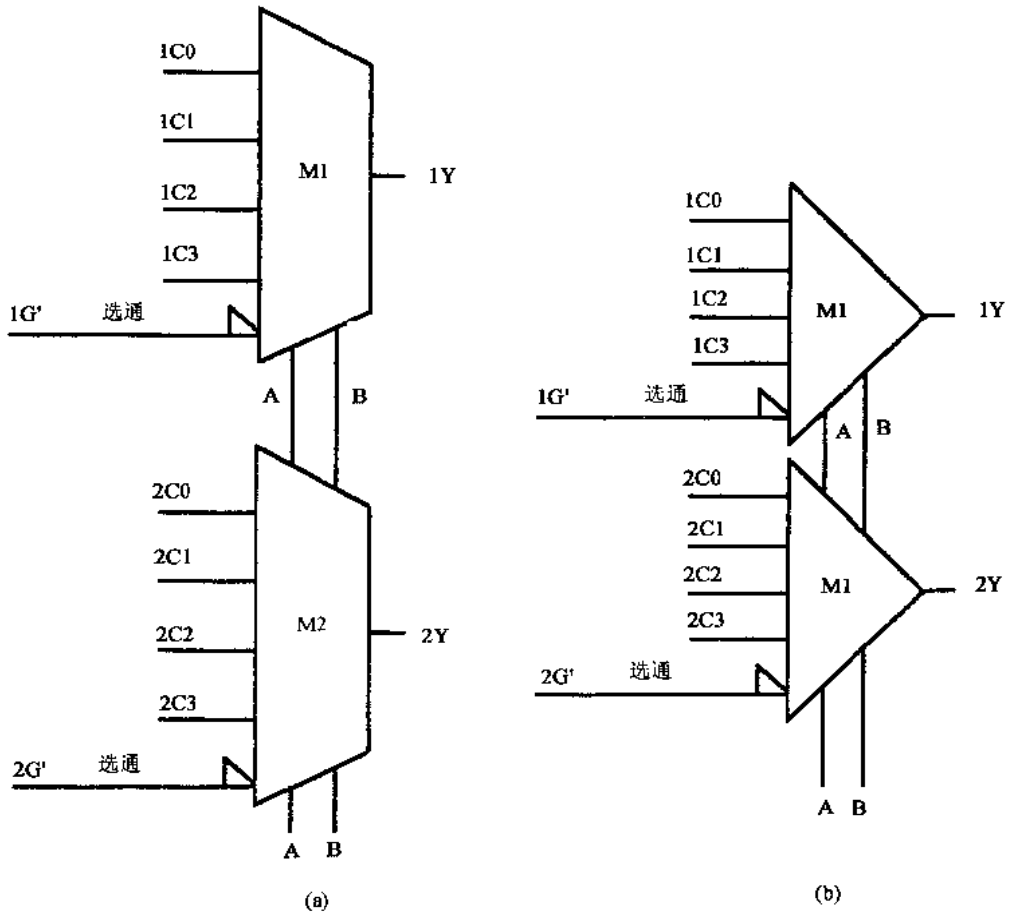


图 B-5　有选通端的双 4 线数据多路选择器

对于同样的电路 IEEE/IEC 符号表示在图 B-6,公用的选择信号输入端表示在符号的惟一变形部分,选通输入端标着“EN”,代表“使能”。

触发器(例,74×78 双 JK 触发器)

这个元件包含两个 JK 触发器,有单独的可预置输入端,其内部有联在一起的公用清零和时钟线,图 B-7 表示的是广泛使用的通用触发器符号。在图 B-8 的 IEEE/IEC 符号中,代表特性的变形“方框”为公用信号输入出现在图顶,象多路选择器那样,还要注意触发器输出“右角”上的符号,表示这些输出是被寄存的。

B.4　IEC 符号扩展到更复杂的电路

移位寄存器(例,74×163 并行输入 8 位移位寄存器)

IEC 符号可直截了当地扩展到更复杂的电路,如图 B-9 所示。象前面讨论过的一样,寄存

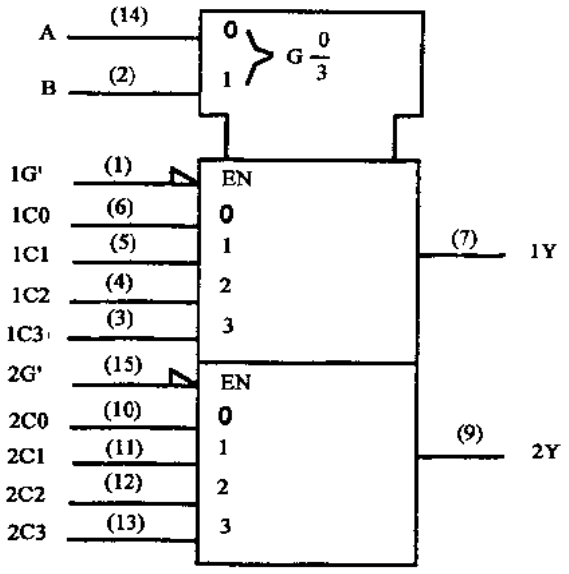


图 B-6　有选通端的双 2 选择输入多路选择器

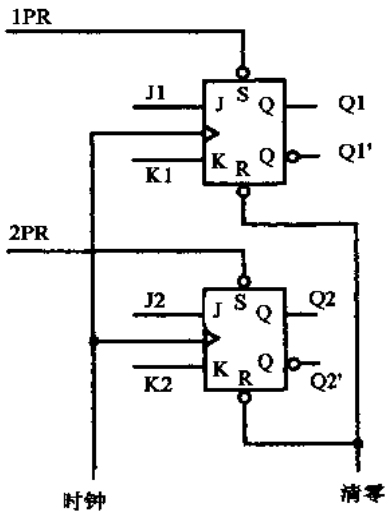


图 B-7

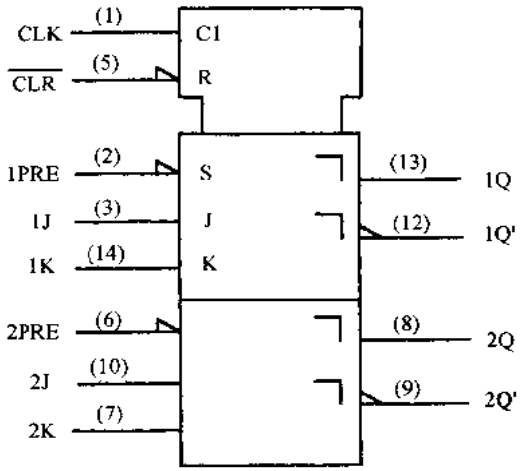


图 B-8

器各级的输入公用信号加在图顶上惟一变形的方框上。

公用清零信号是有源低电平有效(半箭头),并连接在 9 端。移位/输入信号控制工作方式,当此信号为高电平时,时钟脉冲引起移位发生,当此信号为低电平时,并行输入数据。两个时钟输入被'或',若它们中的一个或两个保持有效(高),定时时钟就被禁止,这样有一个时钟线能用作同步脉冲,而另一个起使能控制作用。

有一个 8 级移位寄存器,每级用一个矩形表示,并行数据输入用字母 A 到 H 表示,并且移位寄存器各级按字母顺序连接。A 区还有一个输入(SER),如果寄存器要求串行输入数据,则使用此输入(SER)信号端。只是在最后一级(H 区)才有一个输出端用作串行数据输出。

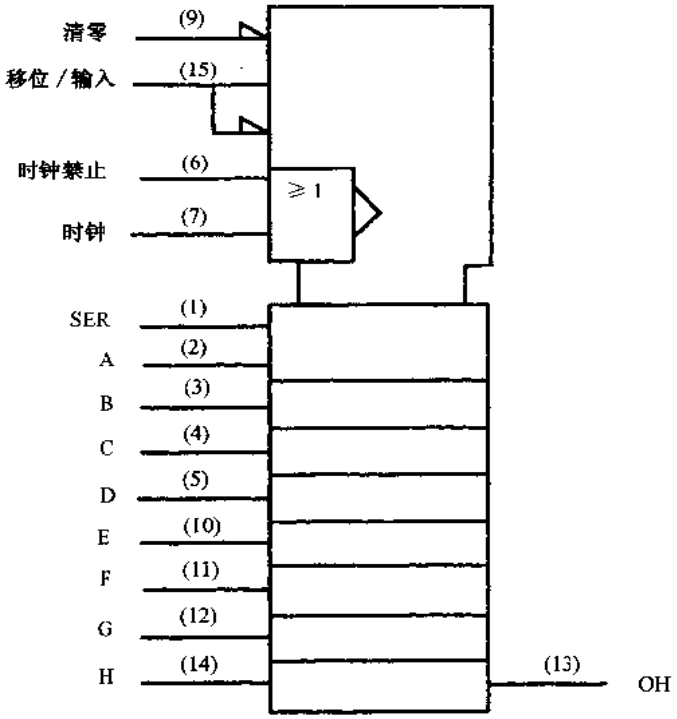


图 B-9

计数器(例,74×160 同步 4 位十进制计数器)

参考图 B-10,IEEE/IEC 的符号,该器件有几个公用输入端,分别连接到计数器的 4 级上,除公用时钟外,还有一个公用清零输入端(CLR)是有源低电平有效。一个公用的方式-控制信号输入端(LOAD),当它为低电平时,选择方式 1(M1)为并行数据输入,当它为高电平时,选择计数方式(M2)。还有两个公用高电平有效使能端(ENT 和 ENP),两个输入必须全是高电平时,进入计数状态。ENT 单独使能,则有波纹进位输出(RCO),4 级计数器中的每一级都有数据输入端和一个输出端(标着 Q)。

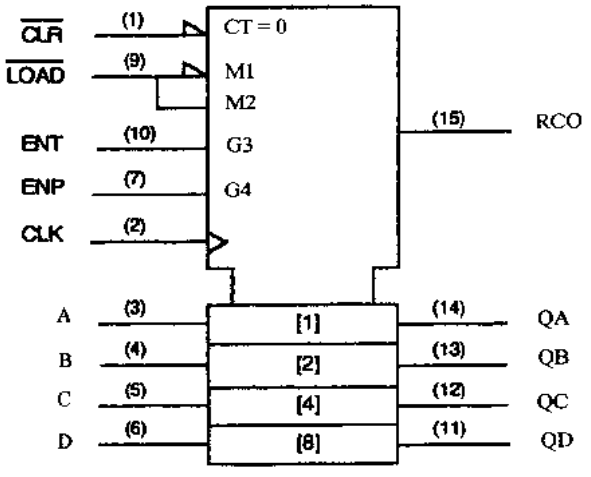


图 B-10

附录 C　数字逻辑模拟的一些说明

C.1　有反馈电路模拟中的问题

使用计算机模拟测试逻辑电路是一个强有力的工具，但是易犯错误，如果未认识到，可能会产生严重的后果。例如，考虑图 7-8 门控锁存器的设计，LogicWorks™屏幕图像重新画在图 C-1。

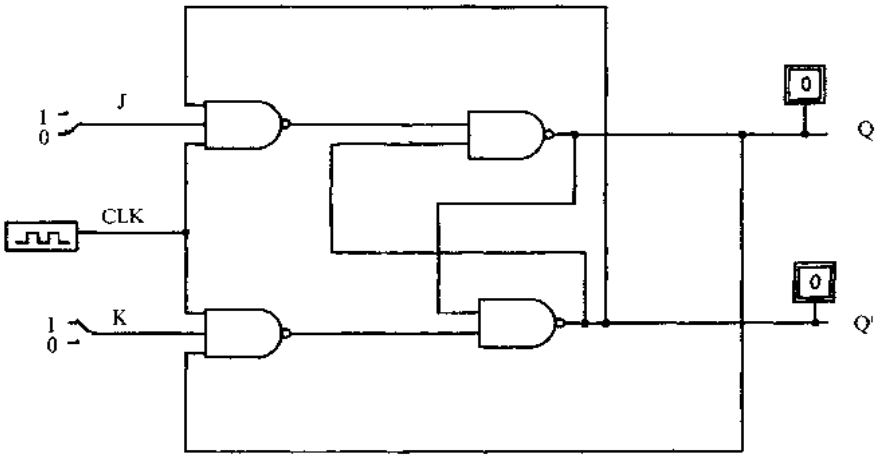


图 C-1

象 7.5 节讨论的那样，希望有振荡条件产生，但所做的结果成了图 C-2 表示的那样注意不管用输入开关怎么做都不会结束振荡，即使复位定时时钟回到零再从新启动它也要持续振荡，更坏的情况是停止电路工作重新启动程序，这个问题仍旧存在——甚至部分损坏也还是这样。

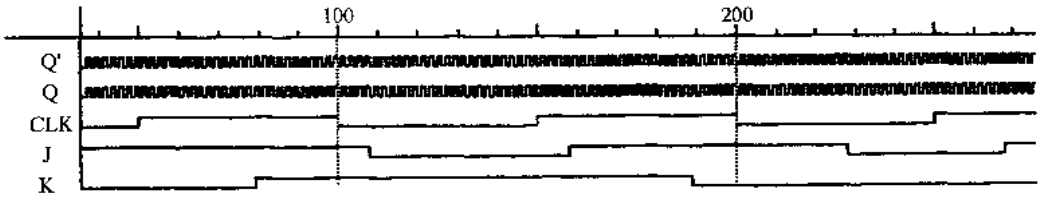


图 C-2

事实上问题出自仿真器象是逻辑工作状态，为了完成适当的功能，需要从一定的条件启动，这个条件是所有门的状态是可确定的。交叉耦合锁存(许多其他有反馈连接的)电路可能对一定的输入组合，有两个能同等程度地发生的输出状态。这样混乱就会发生。不可能的行为会发生很强的错误征兆，如锁存器输出两个零(图 C-1)，或者反相器有相同的输入和输出逻辑电平。象实际电路测试那样，工程师一定不能太轻信；示波器甚至计算机都可能误导我们。

解决办法

复位电路应该加上，最合适的方法是在关键反馈路径上强迫这个电路有惟一的状态。在本例中，加上单与门和开关(图 C-3)将会满意地解决。复位门输入端应该做到无时间延迟，以致于当它切换到逻辑 1 时，使整个电路工作不会发生紊乱。

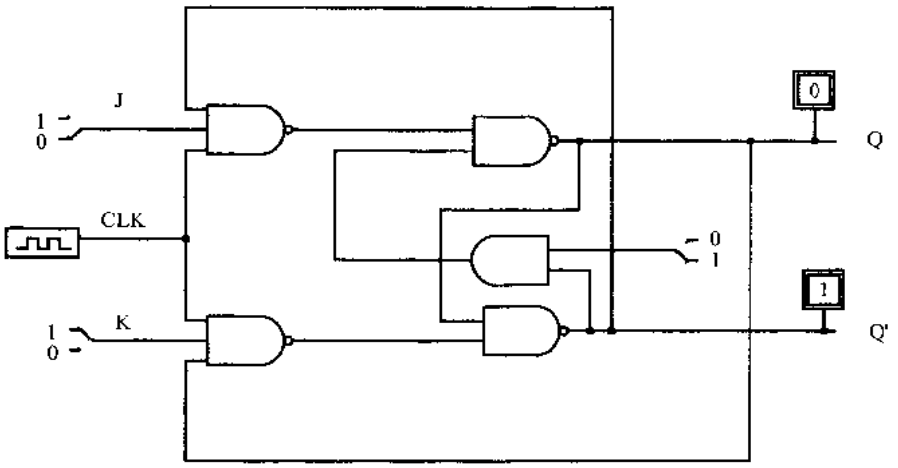


图 C-3

C.2　时钟同步化输入波形的产生

在练习题 7.8 中，输入信号起始于初值逻辑 1，要求在每 4 个时钟脉冲的下降沿改变电平，这在模拟中可能是容易完成的，用两个时钟源(每个波形为一个时钟源)，像所要求的那样，调整好每个时钟源通和断的时间。在本例情形中，时钟已被任意设为 10 个时间单元为高电平和 40 个时间单元为低电平，随后定义输入信号源有相等的高、低电平为 200 个时间单元。

练习题 8.21 中在重新给出了有重要改变的电路结构图，象练习题 7.8 解答中指出的那样，模拟信号瞬间产生零建立和保持时间的情况，使触发器 A 不可能在下一个连续的时钟脉冲到来之前对输入端的变化响应。如果避免这个条件，输入信号能与时钟前沿同步代替它的后沿，在仿真器中通过加入 10 个时间单元对时钟信号延迟，有效地向右移，响应它的前沿对准输入信号跳变，象图 8-39 所示。两个反相器使用的目的是没有反相。

练习题 7.8 要求一个初始值为 1 的输入波形，在练习题 8.21 中，在 LogicWorks™仿真器上能容易地完成，当模拟被暂停时，在定时图窗口人为地画出初始逻辑高电平。作者已选择用 INPUT = 0 代替起始状态，因为它允许定时图的重要特性更容易适应图 8-40 中得到。